

С.М.Львовский
НАБОР И ВЕРСТКА В ПАКЕТЕ TEX

Книга посвящена популярной издательской системе LATEX, предназначеннай для набора и верстки сложных научно-технических текстов с большим количеством формул, таблиц, диаграмм и пр. Во втором издании обновлен практически весь материал книги и добавлены сведения о работе в новой версии — LATEXe 2ε. Кроме того даны 5 новых приложений: это — импорт графики, программа makeindex, шрифты Американского математического общества, пакет AMS-LATEX, и информация о том, где все это можно получить.

Полезна всем, кто имеет дело с изготовлением на компьютере оригинал-макетов изданий, а также авторам, самостоятельно набирающим на компьютере свои научные работы.

Оглавление

Предисловие	9	3.5. Корни	31
I Элементарное введение	11	3.6. Штрихи и многоточия	31
1. Общие замечания	11	3.7. Функции типа синус	32
1.1. Что такое TEX и LATEX	11	4. Разбиение исходного файла на	32
1.2. Достоинства и недостатки	12	части	
1.3. Литература по TEXу	14	5. Обработка ошибок	34
1.4. Как проходит работа с	14	6. Как, читать книгу дальше?	41
системой LATEX		II Как набирать формулы	43
2. Основные понятия	15	1. Таблицы спецзнаков с	43
2.1. Исходный файл	15	комментариями	
2.2. Спецсимволы	16	1.1. Операции, отношения и	43
2.3. Команды и их задание в	17	просто значки	
тексте		1.2. Операции с пределами и без	46
2.4. Структура исходного текста	19	разделов	49
2.5. Группы	20	1.3. Разное	49
2.6. Параметры	22	1.4. Какие еще есть символы	51
2.7. Команды с аргументами	22	2. Важные мелочи	51
2.8. Окружения	24	2.1. Нумерация формул	51
2.9. Звездочка после имени	25	2.2. Переносы в формулах	53
команды		2.3. Смена шрифтов в формуле:	54
2.10. Единицы длины	25	LaTEX 2.09	
2.11. Автоматическая генерация	26	2.4. Смена шрифтов в формуле:	56
ссылок		LATEX 2ε	
3. Набор формул в простейших	28	2.5. Включение текста в	58
случаях		формулы	
3.1. Основные принципы	28	2.6. Скобки переменного	59
3.2. Степени и индексы	28	размера	
3.3. Дроби	29	2.7. Перечеркнутые символы	62
3.4. Скобки	30	2.8. Надстрочные знаки	63
		2.9. Альтернативные	64

обозначения для математических формул		разрывами строк	
3. Одно над другим	65	7.4. Верстка абзацев без выравнивания и переносов	111
3.1. Простейшие случаи	65	7.5. Более тонкая настройка	111
3.2. Набор матриц	67	8. Между абзацами	116
3.3. Набор коммутативных диаграмм в LATEX'e 2e	73	8.1. Понятие о режимах TEX'a	116
3.4. Переносы в выключочных формулах: LATEX 2.09	75	8.2. Подавление абзацного отступа	117
4. Тонкая настройка	76	8.3. Управление разрывами страниц	118
4.1. Пробелы вручную	76	8.4. Вертикальные промежутки	121
4.2. Размер символов в формулах	77	8.5. Интерлиньяж	123
4.3. Фантомы и прочее	79	8.6. Набор в две колонки	124
4.4. Снова об интервалах в формулах	81	8.7. Заключительные замечания о разрывах страниц и вертикальных интервалах	125
4.5. Дополнительные пробелы вокруг формул	84	9. Специальные абзацы	125
III Набор текста	86	9.1. Цитаты	126
1. Специальные знаки	86	9.2. Центрирование, ровнение текста по краю	126
1.1. Дефисы, минусы и тире	86	9.3. Стихи	128
1.2. Кавычки	87	9.4. Перечни	128
1.3. Многоточие	88	9.5. Буквальное воспроизведение (<i>verbatim</i> , <i>verb</i>)	133
1.4. Прочие значки	88	9.6. Абзацы нестандартной формы	134
1.5. Подчеркивания, рамки	89	10. Линейки	137
2. Промежутки между словами	89	10.1. Линейки в простейшем виде	137
2.1. Неразрывный пробел	89	10.2. TEX'овские команды для генерации линеек	138
2.2. Промежутки между предложениями	90	10.3. Невидимые линейки	139
2.3. Установка промежутков вручную	91	IV Оформление текста в целом	141
3. Диакритические знаки и прочее	92	1. Стили и стилевые опции в LATEX'e 2.09	141
4. Смена шрифтов в тексте: LATEX 2.09	94	2. Классы документов и их опции в LATEX'e 2e	143
5. Смена шрифтов в тексте: LATEX 2e	98	3. Деловые письма на LATEX'e	146
6. Сноски	101	4. Стиль оформления страницы	148
7. Абзацы	102	5. Поля, размер страницы и прочее	149
7.1. Переносы	105	5.1. Ширина	149
7.2. Команда \sloppy и параметр \emergencystretch	107		
7.3. Ручное управление	108		

5.2. Высота	151	4.2. Расширение возможностей array и tabular	204
5.3. Сдвиг страницы как целого	152	VII Создание новых команд	209
6. Разделы документа	152	1. Макроопределения	209
6.1. Команда \section	152	1.1. Команды без аргументов	209
6.2. Какие бывают разделы документа	153	1.2. Команды с аргументами	215
6.3. Изменение стандартных заголовков	154	1.3. Новые возможности LATEX'a 2ε	218
6.4. До и после основного текста	155	2. Счетчики	219
7. Титул, оглавление и пр.	156	2.1. Создание счетчиков и простейшие операции с ними	220
7.1. Титульный лист	156	2.2. Отношение подчинения между счетчиками	223
7.2. Оглавление	158	2.3. Организация автоматических ссылок	226
7.3. Список литературы	158	2.4. Счетчики, которые определять не надо	231
7.4. Предметный указатель	160	2.5. Модификация оформления перечней	231
7.5. Перемещаемые аргументы и хрупкие команды	164	3. Параметры со значением длины	233
8. Плавающие иллюстрации и таблицы	165	4. Создание новых окружений	236
8.1. Нештатные ситуации с плавающими объектами в LATEX'e 2.09 и в LATEX'e 2ε	168	4.1. Новые окружения: общий случай	236
9. Заметки на полях (маргиналии)	170	4.2. Окружения типа "теорема"	239
V Псевдорисунки	172	VIII Блоки	242
1. Создание псевдорисунка	172	1. Текст состоит из блоков	242
2. Отрезки и стрелки	175	2. LATEХовские команды для генерации блоков	243
3. Окружности, круги и овалы	176	2.1. Блоки из строки	243
4. Кривые (LATEX 2ε)	177	2.2. Блоки из абзацев	245
5. Дополнительные возможности	178	2.3. Текст в рамке; комбинации блоков	247
6. Параметры оформления псевдорисунка	181	2.4. Сдвиги относительно базисной линии	248
VI Верстка текста с выравниванием	182	2.5. Блоки: дополнительные возможности LATEX 2ε	250
1. Имитация табулятора	182	3. Команда \hbox	252
1.1. Элементарные средства	182	3.1. Растижимые интервалы	253
1.2. Более сложные средства	185	3.2. Лидеры	255
2. Верстка таблиц	188	3.3. Клей	257
2.1. Простейшие случаи	188	3.4. Бесконечно сжимаемые интервалы	259
2.2. Более сложные случаи	191		
3. Примеры	195		
4. Дополнительные возможности LATEX'a 2ε	203		
4.1. Пересечения линеек	203		

3.5. Еще раз о линейках	261	пакете emTEX	
4. Команда \vbox	262	Приложение Г. Программа	323
5. Блоковые переменные	264	makeindex	
IX Модификация стандартных стилей	267	1. Простейшие средства	323
1. Общие замечания	267	2. Тонкости	326
2. Снова о счетчиках	268	3. Настройка программы	327
3. Разделы документа	272	makeindex	
3.1. Что нумеровать и что включать в оглавление	272	Приложение Д. Шрифты AMS	330
3.2. Модификация команд, задающих разделы	273	Приложение Е. AMS-LATEX в	335
4. Оглавление, список иллюстраций и прочее	279	рамках LATEX'a 2 ϵ	
5. Перечни общего вида	286	1. Матрицы (пакет amsmath)	336
5.1. Параметры, влияющие на оформление перечней	287	2. Маленькие хитрости (пакет amsmath)	337
5.2. Окружения list и trivlist	288	3. Дроби и их обобщения (пакет amsmath)	340
6. Колонтитулы	290	4. Выключные формулы (пакет amsmath)	341
7. Плавающие объекты	300	5. Работа с теоремами (пакет amsthm)	344
8. Разное	304	6. AMS-LATEX'овские классы	345
8.1. Теоремы, выключные формулы	304	документов	
8.2. Сноски	306	Приложение Ж. Откуда взять TeX?	347
8.3. Список литературы	308	1. Электронный дом для TgX'a	348
8.4. Предметный указатель	308	2. Ассоциации пользователей	350
Приложение А. О TEX'овских шрифтах	311	TgX'a и проблемы	
Приложение Б. LATEX и другие макропакеты	315	русификации	
Приложение В. Импорт графики в	319	Литература	353

Предметный указатель

\ (backslash с пробелом)	18, 91	\+187
\" 92		\, 76,87
_ 16		\[65
\# 16		\] 65
\! 70,76		\(64
\\$ 16		\) 64
\% 16		\{ 16, 30,60
\` 92		\} 16, 30,60
в окружении tabbing	186	

* 110
\ 68, 109, 183, 189
в абзаце 109
в окружении array 68
в окружении tabbing 183
в окружении tabular 189
\- 106
в окружении tabbing 187
для предотвращения переносов
109
\. 92
\| 95
\: 76
\; 76
\< 187
\=92
в окружении tabbing 182
\^92
\` (открывающийся апостроф) 92
\~ 92
\> 182
\@ 91
\! 93
\? 93
\addtoreset 269
\afterindentfalse 279
\afterindenttrue 279
\begintheorem 304
\biblabel 308
\chapapp 277
\cite 308
\dotsep 283
\dottedtocline 282
\endtheorem 305
\eqnnumi 305
\evenfoot 291
\evenhead 291
\idxitem 308
\makechapterhead 277
\makefnmark 307
\makefntext 307
\makeschapterhead 278
\oddfoot 291
\oddhead 291
\opargbegintheorem 304
\pnumwidth 283
\startsection 273
\tocrmarg 283
\AA 93
\AE 93
\Acute 339
\Alph 221
\Bar 339
\Bbb (AMS-TEX) 316
\Bbbk 333
\Biggl 62
\Biggr 62
\Bigl 62
\Bigr 62
\Box 50
\Breve 339
\Buinpeq 331
\Cap 330
\Check 339
\Cup 330
\Ddot 339
\DeclareHathOperator 339
\DeclareMathOperator* 339
\Delta 44
\Diaaond 50
\Dot 339
\Downarrow 46
\EuScript 58
\Finv 333
\Game 333
\Gamma 44
\Grave 339
\H 92
\Hat 339
\Huge 97
\Im 50
\Join 45
\L 93
\LARGE 97

\LaTeX 18
\LaTeXe 18
\Lambda 44
\Large 97
\Leftarrow 46
\Leftrightarrow 46
\leftarrow 333
\Longleftarrow 46
\Longleftrightarrow 46
\Longrightarrow 45
\Lsh 333
\O 93
\OE 93
\Omega 44
\P 50
\Phi 44
\Pi 44
\Pr 48
\Psi 44
\Re 50
\Rrightarrow 45
\Roman 221
\Rrightarrow 333
\Rsh 333
\S 50, 88
\Sigma 44
\Subset 331
\Supset 331
\TeX 18
\Theta 44
\Tilde 339
\Uparrow 46
\Updownarrow 46
\Upsilon 44
\Vdash 331
\Vec 339
\Vert 51
\Vvdash 331
\Xi 44
\' в окружении tabbing 187
\a 184, 186
\aa 93
\abstractname 155
\acute 63
\addcontentsline 281
\address 146, 345
\addtocontents 280
\addtocounter 220
\addtolength 235
\ae 93
\afterpage 170
\aleph 49
\alph 221
\alpha 43
\amalg 44
\and 157
\angle 49
\appendix 156
\appendixname 155
\approx 45
\approxeq 332
\arabic 221
\arccos 47
\arcsin 47
\arctan 47
\arg 46
\arraycolsep 199
\arrayrulewidth 199
\arraystretch 201
\ast 51
\asubjclass 346
\asymp 45
\atop 66
запрещена в AMS-LATEX'e 340
\author 157
в AMS-LATEX'e 345
\b 92
\backepsilon 332
\bactanatter 156
\backprime 331
\backsimeq 331
\backslash 50, 60

\bar{63}
\barwedge 330
\baselineskip 120, 151
\baselinestretch 124
\batchmode 41
\beta 43 \beth 333
\between 331
\bf 20, 96
 в формулах 54
\bfseries 99
\bibitem 159
 с необязательным аргументом 160
\bibname 155
\bigcap 48
\bigcirc 44
\bigcup 48
\biggl 62
\biggr 62
\bigl 62
\bigodot 48
\bigoplus 48
\bigotimes 48
\bigr 62
\bigskip 121
\bigskipamount 110
\bigsqcup 48
\bigstar 333
\bigtriangledown 44
\bigtriangleup 44
\biguplus 48
\bigvee 48
\bigwedge 48
\binom 340
\binoppenalty 53, 54
\blacklozenge 333
\blacksquare 333
\blacktriangle 333
\blacktriangledown 333
\blacktriangleleft 331
\blacktriangleright 331
\bold AMS-TEX 316
\boldmath 55
\bot 50
\bottigrule 303
\bottomfraction 301
\bowtie 45
\boxdot 330
\boxed 340
\boxminus 330
\boxplus 330
\boxtlines 330
\breve 63
\bullet 44
\buapeq 331
\c 92
\cal 55
\cap 44
\caption 166
 с необязательным аргументом 168
\cases (Plain TEX) 317
\cc 147
\ccname 147
\cdot 44
\cdots 31
\centerdot 330
\cfrac 338
\chapter 153
\chaptemame 154
\check 63
\checkmark 334
\chi 43
\choose 66
 запрещена в AMS-LATEX'e 340
\circ 44
\circle 176
\circle* 176
\circlearrowleft 332
\circlearrowright 332
\circledR 334
\circledS 333
\circledast 330
\circledcirc 330
\circleddash 330
\cite 159

с необязательным аргументом 159
\cleardoublepage 119
\clearpage 119, 169
\cline 192
\closing 146
\clubsuit 50
\colon 81, 82
\columnsep 150
\columnseprule 150
\complement 333
\cong 45
\contentsname 155, 286
\coprod 48
\copy 266
\copyright 50, 88
\cos 47
\cosh 47
\cot 47
\coth 47
\cr 316, 317
\csc 47
\cup 44
\curlyeqprec 331
\curlyeqsucc 331
\curlyvee 330
\curlywedge 330
\curraddr 345
\curvearrowleft 333
\curvearrowright 333
\d 92
\dag 50
\dagger 44
\daleth 333
\dasharrow 333
\dashleftarrow 333
\dashrightarrow 334
\dashv 45
\date 157
 в письме 146
\dbinom 340
\dblfigrule 303
\dblfloatpagefraction 302
\dblfloatsep 302
\dbltextfloatsep 302
\dbltopfraction 302
\ddag 50
\ddagger 44
\ddot 63
\ddots 68
\dedicatory 346
\deg 47
\delta 43
\det 48
\frac 340
\diagdown 331
\diagup 331
\diamond 44
\diamondsuit 50
\digamma 333
\dim 46
\displaystyle 78
\div 44
\divideontimes 330
\documentclass 20, 143
\documentstyle 19, 141
\dot 63
\doteq 45
\doteqdot 334
\dotfill 255
\dotplus 330
\dots 88
 в AMS-LATEX'e 338
\doublebarwedge 330
\doublecap 334
\doublecup 334
\downarrow 46
\downdownarrows 332
\downharpoonleft 333
\downharpoonright 333
\ell 50
\em94
\email 345
\emergencystretch 108, 113
\emph 100

\emptyset 49
\enclname 147
\endinput 33
\enlargethispage 120
\enskip 91
\ensuremath 212
\epsilon 43
\eqcirc 331
\eqno 52
\eqref 338
\eqsim 332
\eqslantgr 331
\eqslantless 331
\equiv 45 \eta 43 \eth 333
\evensidemargin 150
\exhyphenpenalty 115
\exists 49
\exp 47
\extrarowheight 205
\allingdotseq 331
\fbox 89
\fboxrule 247
\fboxsep 247
\figurename 155, 166
\fill 122, 259
\firsthline 205
\flat 50
\floatpagefraction 301
\floatsep 302
\flushbottom 125
\fnsymbol 222
\footnote 101
\footnotemark 102
\footnoterule 306
\footnotesep 307
\footnotesize 97
\footnotetext 102
\footskip 292
\forall 49
\frac 30
\frak (AMS-TEX) 316
\framebox 247
 B LATEX'e 2ε 251
\frenchspacing 90
\frontmatter 156
\frown 45
\fussy 107
\gainma 43
\gcd 48 \ge 29, 45
\genfrac 340, 341
\geq 51
\geqq 331
\gaqslant 46, 331
\gets 46
\gg45
\ggg331
\gggr 334
\gimel 333
\gnapprox 331
\gneq 331
\gneqq 331
\gnsim 331
\grave 63
\grdot 330
\gtreqless 331
\gtreqqless 331
\gtrless 331
\gvertneqq 331
\hangafter 135
\hangindent 135
\hat 63
\hbar 49
\hbox 252
\hdotsfor 336
\headheight 151, 292
\headsep 151, 292, 293
\heartsuit 50
\height 251
\hfil 254
\hfill 255
\hfuzz 112
\hhline 203
\hline 190
\hoffset 152

\hom 47
\hookleftarrow 46
\hoolrightarrow 45
\phantom 80
\hrule 138
\rulefill 255
\hslash 333
\hspace 91, 123
\hspace* 92
\hss 259
\huge 97
\hyphenation 106
\hyphenpenalty 115
\i 93
\iiint 339
\iiint 339
\iint 339
\imath 50
\in 45
\index 161
\indexentry 162
\indexname 155
\indexspace 161
\inf 48
\infty 49
\input 32
\int 49
\intercal 330
\intertext 343
\intextsep 302
\iota 43
\it96
\item 128
 в окружении theindex 161
 квадратная скобка после команды
 130
 необязательный аргумент 129,132
\itemsep 288
\itshape 99
\j 93
\jmath 50
\kappa 43
\ker 46
\keywords 346
\kill 183
\l 93
\label 26
\labelenuini 232
\labelenunii 232
\labelenumiii 232
\labelenuniv 232
\labelitemi 231
\labelitemii 231
\labelitemiii 231
\labelitemiv 231
\labelsep 287
\labelwidth 287
\lambda 43
\land 51
\angle 60
\large 97
\lasthline 205
\lbrace 51
\lbrack 51
\lceil 60
\ldotp 81, 82
\ldots 31, 88
\le 29,45
\leaders 255, 266
\leadsto 46
\left 30, 59
\leftarrow 51
\leftarrowtail 333
\lefteqn 72, 80, 261
\leftharpoondown 46
\leftharpoonup 46
\leftleftarrows 332
\leftmargin 287
\leftmark 294
\leftrightarrow 46
\leftrightarrows 333
\leftrigharpoons 332
\leftskip 284
\lettbreetimes 330

\leq 51
\leqno 53
\leqq 331
\eqslant 46, 331
\lessapprox 331
\lessdot 330
\lesseqgtr 331
\lesseqqgtr 331
\lessgtr 331
\lessim 331
\lfloor 60
\lg46
\hd 45
\lim 48
\liminf 48
\limits 49
\limsup 48
\line 175
\linebreak 109
 с необязательным аргументом 110
\line-thickness 181
\listfigurename 155
\listoffigures 168
\listoftables 168
\listparindent 288
\listtablename 155
\ll 45
\lap 174
\lcorner 334
\lil 331
\lless 334
\ln 46
\lnapprox 331
\lneq 3.11
\lneqq 331
\lnot 51
\lein 331
\log 46
\longleftarrow 46
\longleftrightarrow 46
\longmapsto 46
\longrightarrow 45
 с необязательным аргументом 110
\looseness 114
\lor 51
\lowercase 296
\lozenge 333
\lrcorner 334
\ltimes 330
\lvertneqq 331
\magstep 313
\magstephalf 313
\mainmatter 156
\makeatletter 268
\makeatother 268
\makebox 244
 в LATEX'e 2ε 250
\makeindex 161
\makelabel 8 147
\maketitle 156
\maltese 334
\mapsto 46
\marginpar 170
 с необязательным аргументом 170
\marginparpush 171
\marginparsep 171
\marginparwidth 171
\markboth 293
\markright 293, 295
\mathbb 57
\mathbf 56
\mathbin 82
\mathcal 56
\mathfrak 57
\mathit 56
\mathop 82
\mathrel 82
\mathrm 56
\mathsf 56
\mathstrut 79
\mathsurround 84, 232, 307
\mathtt 56
\matrix (Plain TEX) 316
\max48
\mbox 109, 243

в формулах 58
для предотвращения переноса 109
как «пустой текст» на странице 119

\mdseries 99
\measuredangle 333
\medskip 121
\medskipamount 110
\mho 50, 333
\mid 45
\min 48
\mit 55
\mod 339
\models 45
\mp44
\mu43
\multicolumn 192
\multiput 178, 265
\multilinegap 342
\nLeftrightarrow 333
\nLeftarrow 333
VnRightarrow 333
\nVDash 332
\nVdash 332
\nabla 49
\natural 50
\ncong 332
\ne45
\nearrow 46
\neg 49
\neq 51
\newcommand 210, 218
\newcounter 220
 с необязательным аргументом 224
\newenvironment 237
\newenvironment* 238
\newfont 311
\newlength 233
\nevpage 119
\newtheorem 239
\newtheorem* 344
\exists 333
\neq 332
\geqq 332
\geqslant 332
\ntr 332
\ni45
\leftarrow 333
\nleftrightarrow 333
\leq 332
\leqslant 332
\less 332
\nmid 332
\noindent 117
\nolimits 49
\nolinebreak 110
\nonfrenchspacing 90
\nonstopmode 41
\nonumber 71
\nopagebreak 118
\normalfont 101
\normalniarginpar 171
\normalsize 97
\not 62
\notag 342, 343
\notin 45
\parallel 332
\prec 332
\preceq 332
\rightarrow 333
\shortmid 332
\shortparallel 332
\sim 332
\subseteq 332
\subseteqq 332
\succ 332
\succcurlyeq 332
\supseteq 332
\supseteqq 332
\triangleleft 332
\trianglelefteq 332
\triangleright 332
\trianglerighteq 332
\nu43

\numberwithin 338
\nvDash 332
\nvdash 332
\nwarrow 46
\o 93
\oddsidemargin 150
\odot 44
\oe 93
\of 315
\of (Plain TEX) 316
\oint 49
\omega 43
\ominus 44
\onecolumn 125
\opening 146
\oplus 44
\oslash 44
\otimes 44
\oval 176
 с необязательным аргументом 176
\over 315
\over (Plain TEX) 315
\overbrace 67
\overleftarrow 64
\overline 63
\overrightarrow 64
\owns 51
\pagebreak 119
\pagename 145
\pagenumbering 148
\pageref 27
\pagestyle 148, 292
\par 117
\paragraph 153
\parallel 45
\parbox 245
 в LATEX'e 2ε 251
 с необязательным аргументом 246
\parindent 22
\parsep 288
\parshape 136
\parskip 124
\part 153
\partial 49
\partname 155
\partopsep 288
\perp 45
\phantom 80
\pbi 43
\pi43
\pitchfork 331
\pm44
\pod 339
\poptabs 186
\pounds 50, 88
\prec 45
\precapprox 332
\preccurlyeq 331
\preceq 45
\precnapprox 331
\precneqq 331
\precsim 331
\precsim 331
\prime 49
\proclaim (Plain TEX) 318
\prod 48
\rootname 344
\proto 45
\protect 164, 280
 в аргументе
 \addtocontents 280
 в аргументе
 \addcontentsline 282
 в пометке 299
\ps 147
\psi 43
\pushtabs 186
\put 173
\qbezier 177
\qqquad 58, 75, 76, 91
\quad 76, 77, 91
\r 92
\raggedbottom 125
\raggedright 111

\raisebox 248
\rangle 60
\brace 51
\brack 51
\ceil 60
\ref 27
 в окружении enumerate 131
 ссылка на плавающую
 иллюстрацию 166
 ссылка на счетчик, определенный
 пользователем 226
 ссылки на раздел документа 153
 ссылки на формулы 52
\refname 35
\refstepcounter 225
\relax 139
\relpenalty 54
\renewcoamand 145, 147, 155, 214
\renewcommand* 218
\renewenvironment 238
\reversemarginpar 171
\rfloor 60
\rhd 45
\rho 43
\right 30, 59
\rightarrow 51
\rightarrowtail 333
\rightharpoondown 46
\rightharpoonup 46
\righthyphenmin 105
\rightleftarrows 333
\rightleftharpoons 46, 332
\rightmargin 287
\rightinark 294
\rightrightarrow 332
\rightkip 284
\rightsquigarrow 332
\righttureet lines 330
\risingdotseq 331
\rlap 260
\rm 21, 96
 в формулах 54
\rmfamily 99
\roman 221
\roman (AMS-TEX) 316
\root 315
\root (Plain TEX) 316
\times 330
\rule 137
 с необязательным аргументом 137
\samepage 119
\savebox 266
\sbox 265
\sc96
\scrptscriptstyle 78
\scriptsize 97
\scriptstyle 78
\scrollmode 41
\scshape 99
\searrow 46
\sec 47
\section 152, 273
 в AMS-LATEX'e 346
 с необязательным аргументом 152
\section* 153
\sectionmark 295
\setcounter 220
\setlength 234
\setminus 44
\settodepth 235
\settoheight 235
\sf 96
 в формулах 54
\sffamily 99
\sharp 50
\shortmid 332
\shortparallel 332
\shoveleft 342
\shoveright 342
\sigma 43
\signature 146
\sim 45
\simeq 45
\sin 47

\sinh 47
\sl 18, 96
 в формулах 54
\loppy 107, 113
 в LATEX'e 2ε 107, 114
\slshape 99
\small 97
\sinalltrovn 331
\smallsetminus 330
\smallskip 121
\smallskipamount 110
\smallsmile 331
\smash 80
\smile 45
\spadesuit 50
\special 172
 в пакете emTEX 319
\specialsection 346
\sphericalangla 333
\sqcap 44
\sqcup 44
\sqrt 31
\sqsubset 45, 331
\sqsubseteq 45
\sqsupset 45, 331
\sqsupseteq 45
\square 333
\ss 93
\stackrel 66
\star 44
\stepcounter 225
\strut 139, 140, 201, 293
\subitem 161, 309
\subparagraph 153
\subsection 153
 в AMS-LATEX'e 346
\subsectionmark 296
\subset 45
\subseteqq 45
\subseteqqq 331
\subsetneq 331
\subsetneqq 331

\Bubsubitem 161, 309
\subsubsection 153
 в AMS-LATEX'e 346
\succ 45
\succapprox 332
\succcurlyeq 331
\succeq 45
\succnapprox 331
\succneqq 331
\succnsim 331
\sum 48
\sup 48
\suppressfloats 169
\supset 45
\supseteq 45
\supseteqq 331
\supsetneq 331
\supsetneqq 331
\surd 50
\swarrow 46
\symbol 88
\t 92
\tabcolsep 199
\tablename 155, 167
\tableofcontents 158
 стандартное определение 286
\tan47
\tanh 47
\tau 43
\binom 340
\text 337
\textbf 99
\textfloatsep 302
\textfraction 301
\textheight 151
\textit 99
\textmd 99
\textnormal 101
\textrm 99
\textsc 99
\textsf 99
\textsl 99

\textstyle 78
\texttt 99
\textup 99
\textwidth 149
\tfrac 340
\thanks 157
 в AMS-LATEX'е 345
\theindex 309
\theoremstyle 344
\theta 43
\thickapprox 332
\thicklines 181
\thicksim 332
\thinlines 181
\thispagestyle 148
\tilde 63
\times 44
\tiny 97
\title 157
 в AMS-LATEX'е 345
\to 45
\tolerance 113
\top 50
\topfigrule 303
\topfr&ction 301
\topmargin 151
\topsep 288
\topskip 151
\totalheight 251
\triangle 49
\triangledown 333
\triangleleft 44
\trianglelefteq 331
\triangleq 331
\angleright 44
\anglerighteq 331
\tt 96
 в формулах 54
\ttfainily 99
\twocolumn 124
\twoheadleftarrowH 332
\twoheadrightarrow 332
\u 92
\uchyph 116
\ulcorner 334
\underbrace 66
\underline 89
\unitlength 173
\unlhd 45
\unrhd 45
\uparrow 46
\updownarrow 46
\upharpoonleft 333
\upharpoonright 332
\uplus 44
\uppercase 296
\upshape 99
\upsilon 43
\upuparrows 332
\urcorner 334
\usebox 265
\usecounter 289
\usepackage 23, 143
 с необязательным аргументом 144
\v 92
\vDaah 331
\value 222
\varDelta 337
\varGamma 337
\varLambda 337
\varOmega 337
\varPhi 337
\varPi 337
\varPsi 337
\varSigma 337
\varTheta 337
\varUpsilon 337
\varXi 337
\varepsilon 43
\varinjlim 339
\varkappa 46, 333
\varliminf 339
\varlimsup 339
\varnothing 50, 333

\varphi 43
\varpi 43
\varprojlim 339
\varpropto 331
\varrho 43
\varsigma 43
\varsubsetneq 331
\varsubsetneqq 331
\varsupsetneq 331
\varsupsetneqq 331
\vartheta 43
\vartriangle 331
\vartriangleleft 331
\vartriangleright 331
\vbox 262
\vdash 45
\vdots 68
\vec{3} 3
\vector 175
\vea 44
\veebar 330
\verb 133
 в макроопределении 211
 в сносках 134
\verb* 134
\voffset 152
\phantom 80
\rule 138, 207
\vspace 121
\vspace* 122
\wedge 44
\widehat{3} 63
\widetilde{3} 64
\width 250
\wp 50
\wr 44
\xi 43
\leftarrow 338
\rightarrow 338
\yen 333
\zeta 43
11pt (стилевая опция) 23, 142
12pt (стилевая опция) 23, 142
a4paper (стилевая опция) 144
a5paper (стилевая опция) 144
abstract (окружение)
в AMS-LATEX'e 346
afterpage (стилевой пакет) 170
align (окружение) 342, 343
align* (окружение) 343
aligned (окружение) 343
amsart (класс документов) 144,
 335, 345
amsbook (класс документов) 144, 335,
 345
amscd (стилевой пакет) 73, 335
amsfonts (стилевой пакет) 23, 57, 330,
 335, 336
AMS-LATEX'e 12, 46, 50, 51, 57,
 73-75, 318, 330-346
amsmath (стилевой пакет) 144, 335
amsproc (класс документов) 144, 335,
 345
amssymb (стилевой пакет) 46, 50, 51,
 330, 335
AMS-TEX 11, 73, 315-318
amsthm (стилевой пакет) 335
array (окружение) 67
array (стилевой пакет) 204
article (стиль в LATEX'e 2.09, класс
 документов в LATEX'e 2 ϵ) 141
at (ключевое слово) 313
at-выражение 198
aux-файл 27
b5paper (стилевая опция) 144
badness 112
bmatrix (окружение) 336
book (стиль в LATEX'e 2.09, класс
 документов в LATEX'e 2 ϵ) 141
bottomnumber (счетчик) 300
CD (окружение) 73
center (окружение) 24, 126
CTAN 348
CyrTUG 351

dbltopnumber (счетчик) 301
definition (стиль оформления теорем в AMS-LATEX'e) 344
depth (ключевое слово) 139
depth (ключевое слово) 261
description (окружение) 128
displaymath (окружение) 65
draft (стилевая опция) 143
dvi-драйвер 15
dvi-файл 15
empty (стиль оформления страниц) 148
emTEX 319, 349
enumerate (окружение) 128, 130
eqnarray (окружение) 70
eqnarray* (окружение) 71
equation (окружение) 52
eufrak (стилевой пакет) 58
euscript (стилевой пакет) 58
executivepaper (стилевая опция) 144
figure (окружение) 165
 с необязательным аргументом 166
figure* (окружение) 167
fleqn (стилевая опция) 143
flushleft (окружение) 126
flushright (окружение) 126
gather (окружение) 342
gather* (окружение) 342
Goossens, M. 12
headings (стиль оформления страниц) 148
height (ключевое слово) 139
hhline (стилевой пакет) 203
idx-файл 162
itemize (окружение) 128, 129
l@-команда 281
landscape (стилевая опция) 145
latextsym (стилевой пакет) 45, 46, 50
lagalpaper (стилевая опция) 144
leqno (стилевая опция) 143
letter (окружение) 146
letter (стиль в LATEX'e 2.09, класс документов в LATEX'e 2 ϵ) 141, 146
list (окружение) 288-290
lot-файл 168, 280
log-файл 34, 103
longtable (стилевой пакет) 23
lot-файл 168, 280
makeindex (программа) 323-329
 стилевой файл 328
math (окружение) 64
matrix (окружение) 336
MaxMatrixCols (счетчик) 337
METAFONT 311
minus (ключевое слово) 257
Mittelbach, F 12
multiline (окружение) 341
multiline* (окружение) 341
myheadings (стиль оформления страниц) 148, 298
notitlepage (стилевая опция) 145
oneside (стилевая опция) 145
openany (стилевая опция) 145
openright (стилевая опция) 145
Overfull 103, 258
 в колонтитуле 153
paragraph (счетчик) 225
picture (окружение) 172
 вложенные окружения 179
pk-файл 311
Plain TEX 11, 315-318
plain (стиль оформления страниц) 148
plain (стиль оформления теорем в AMS-LATEX'e) 344
plus (ключевое слово) 257
pmatrix (окружение) 336
proc (класс документов) 144, 145
proof (окружение) 344
quotation (окружение) 126
quote (окружение) 126

remark (стиль оформления теорем в AMS-LATEX'e) 344
report (стиль в LATEX'e 2.09, класс документов в LATEX'e 2e) 141
scaled (ключевое слово) 312
Schopf, R. 12
secnumdepth (счетчик) 272
subparagraph (счетчик) 225
subsubsection (счетчик) 225
tabbing (окружение) 182
table (окружение) 167
table* (окружение) 167
tabular (окружение) 188-208
 с необязательным аргументом 190
TEXовское приглашение 39
tfm-файл 311
the-команда 227
thebibliography (окружение) 159
theindex (окружение) 160
titlepage (стилевая опция) 143
to (ключевое слово) 253
toc-файл 158, 279
topnumber (счетчик) 300
totalnumber (счетчик) 300
trivlist (окружение) 290
twocolumn (стилевая опция) 142
twoside (стилевая опция) 142
Underfull 104, 258
 при нехватке клея 254
 при печати страницы 125
verbatim (окружение) 133
 в макроопределении 211
 в сносках 134
verbatim (стилевой пакет) 134
verbatim* (окружение) 134
verse (окружение) 128
Vmatrix (окружение) 336
vmatrix (окружение) 336
width (ключевое слово) 139
Абзацы 16,102
 абзацный отступ 22
верстка без выравнивания 111
дополнительный интервал между абзацами 124
изменение количества строк 114
неправильной формы 136
нестандартной формы 134
неточное выравнивание по правому краю 112
подавление отступа 117
сообщения о трудностях при верстке 103, 104
Автор документа 157
Амперсенд 189
Аргумент 23, 57, 93
 необязательный 23
 перемещаемый 165
Базисная линия (baseline) 242
Базовые файлы 350
Бинарные операции 44, 81, 82
Бинарные отношения 45, 81, 82
Биномиальные коэффициенты 66
Блок (box) 242
Блоковые переменные 264
Буквы греческие 43-44
 наклонные 55, 337
Виноградов, Михаил 351
Вулис, Дмитрий 351
Гловти, Нана 351
Группы 20
Дефис 86
Диаграммы 71
Диакритические знаки 92
 в окружении tabbing 184
Длина
 единицы измерения 25
 em 26
 ex 26
 дюйм 25
 пика 25
 пункт 25
 пункт Дидо 25
 цицеро 25
 параметры 22, 233

присваивание значений 234
с коэффициентом 235
сложение 235

Доказательство (AMS-LATEX) 344

Дроби 30

Дроби, цепные 78, 338

Заглавие документа 157

Заметки на полях 170

Иллюстрации 165

- при наборе в две колонки 167
- стандартное название 166

Индексы 28, 29

Интеграл 49

- двойной, тройной, и т. д. 77,339
- контурный 49

Интерлиньяж 123

Кавычки 87

- елочки 87
- лапки 87

Кернинг 87

Клей 122, 257

- бесконечно сжимаемый 259

Кнут, Дональд 11

Колонтитулы 291

- высота 292
- интервал между колонтитулом и текстом 292
- передача информации из текста 293

Команды 17

- примитивные 215
- пробел после имени 18
- со звездочкой 25
- хрупкие (fragile) 165

Комментарии 17

Коммутативные диаграммы 73

Корень 31

Коррекция наклона 95

- в LATEX'е 2ε 100

Лапко, Ольга 351

Лигатуры 87

Лидеры 255

в оглавлении 283

использование блоковых переменных 266

Линейки 137

- невидимые 139,197,201
- в формулах 79

Лэмпорт, Лесли 11

Макросы 209

- новые окружения 237
- с аргументами 216

Маргинации 170

Масштабирование 312

Матрицы 67

- преамбула 68

Маттес, Эберхард 319

Маховая, Ирина 10

Многоточие 88

- в тексте 88
- в формулах 31
- LATEX'е 2ε 338

Надстрочные знаки

- в тексте 92
- в формулах 63
- в пакете amsmath 339

Неразрывный пробел " 89

Оглавление 158

- запись текста без номера страницы 280
- запись текста с номером страницы 281
- модификация оформления 279
- стандартный заголовок 158
- степень детализации 273

Ограничители 60

Окружения 24

- типа «теорема» 239
- в AMS-LATEX'е 344

Операторы типа сумма 48, 81, 82

Опции стилевые 23, 142

- у пакета 144

Пакет, стилевой 23

Переносы

в словах с дефисом 105
в словах, начинающихся с прописной буквы 116
двух последних букв 105
 затруднение и запрет 115
 предотвращение в данном слове с помощью \tbox 109
 с помощью \- 109
 указание разрешенных мест «глобальное» 106
 «локальное» 106
 Перечеркнутые символы 62
 Перечни модификация заголовков enumerate 232
 заголовков itemize 231
 нумерованные (enumerate) 130
 общего вида (list) 288-290
 примитивные (trivlist) 290
 простейшие (itemize) 129
 с заголовками (description) 133
 Пика 25
 Плавающие иллюстрации при наборе в две колонки 167
 стандартное название 166
 Плавающие таблицы 167
 стандартное название 167
 Подчеркивание 89
 Поля 149
 верхнее и нижнее 151
 левое и правое 150
 Пометки (marks) 294
 автоматическое внесение в текст 295
 Преамбула документа 19
 Предметный указатель 160
 модификация оформления 308
 стандартное заглавие 161
 Промежутки
 в формулах 76, 82
 вертикальные 121
 горизонтальные 91, 92
 дополнительные после конца предложения 90
 Пункт 25
 Пункт Дидо 25
 Разделы
 варианты со звездочкой 153
 модификация оформления 273, 277, 279
 подавление отступа в первом абзаце 154, 274
 подавление отступа в первом абзаце главы 279
 стиль оформления заголовка 275
 уровень вложенности 272
 Рамки 89, 247, 248, 340
 Режимы TEX'a 116
 Самарин, Александр 12, 351
 Скобки 30
 горизонтальные 66, 67
 переменного размера 30, 59, 60, 62
 в AMS-LATEX'e 337
 Сноски 101
 к тексту внутри блока 102
 к титльному листу 157
 линейка, отделяющая сноски от текста 306
 оформление номеров 307
 оформление текста сноски 307
 пробел между страницей и сносками 306
 Спивак, Майкл 14
 Список иллюстраций 168
 Список литературы 159
 скобки вокруг номера ссылки 308
 стандартное заглавие 160
 Список таблиц 168
 стандартное заглавие 155
 Сравнения по модулю 47
 Степени 28, 29
 Стихи 128
 Страницы
 запрет разрыва 118

глобальный 119
изменение размера отдельной полосы 120
принудительный разрыв 119
при двустороннем наборе 119
с выдачей плавающих иллюстраций 119
сдвиг как целого 152
стиль нумерации 148
стиль оформления 148
модификация 291
Стрелки 45, 46, 332-333
надпись над стрелкой 66, 74
надпись под стрелкой 74
надпись сбоку от стрелки 72, 74
Строки
жидкие 103
равномерное увеличение разреженности 108
снятие запретов 107
запрет разрыва 110
неразрывный пробел 89
насильственный разрыв 109
с выравниванием 109
Счетчики 219
the-команда 227
выдача на печать 221, 222
определенные при начале трансляции 231
переподчинение существующего счетчика 269, 338
подчинение 224
присваивание значения 220
сложение 220
создание 220
ссылочный префикс 270
увеличение на шаг 225
Таблицы
абзац в графе 193
в пакете array 206
вертикальные линейки 191
горизонтальные линейки 190
графы в несколько колонок 192
интервал между колонками 199
невидимые линейки в строках 201
пreamble 188
!-выражение (в пакете array) 207
at-выражение 198
символ | 191
разбиение preamble на колонки 194
толщина линеек 199
частичные горизонтальные линейки 192
Табуляция
выравнивание по правому краю 186
запоминание и вспоминание позиций 186
использование позиций 182
предварительная установка позиций 183
сдвиг первой позиции 187
установка позиций 182
Танганс 47
Тире 86
длинное (em-dash) 86
короткое (en-dash) 86
Титульный лист 156
дополнительная информация 157
оформление вручную 158
сноски 157
Точка отсчета 174, 242
Ударение 93
Форматный файл 349
Формулы
альтернативные обозначения 64
включение текста 58
внутритекстовые 28, 64
выключочные 28, 65, 143
знак препинания после формулы 31

Формулы

надстрочные знаки 63

нумерация 52, 53, 143, 269, 305

переносы 53, 75

Цепные дроби 78, 338

Цитаты 126

Цицеро 25

Шень, Александр 14, 351

Шрифты

typewriter 96, 314

в формулах 54

капитель 96

курсивный 96

математический курсив 28

наклонный 18, 96

полужирный 20, 96

прямой светлый (roman) 96, 313

рубленый 96

рукописный 55

Штрихи (в формулах) 31, 50, 83, 84

Предисловие

Наверное, каждый, кто впервые знакомится с персональным компьютером, бывает поражен возможностями текстовых редакторов. Но вот первые восторги прошли, и тут обычно обнаруживается по крайней мере два неприятных обстоятельства:

- Качество напечатанного текста невысокое — в лучшем случае как на электрической пишущей машинке.
- Если в тексте встречаются математические формулы, то они либо получаются некрасиво (если в формуле $x + y = z$ не поставить пробелов, то будет тесно, а если поставить, то слишком широко), либо не получаются вовсе, и их приходится вписывать от руки, как в добре старое время.

С этими трудностями отечественные пользователи обычно борются с помощью редактора ChiWriter. Однако даже при использовании его последних версий результат все равно не дотягивает до типографского качества. Существует редактор Word for Windows, позволяющий печатать красивые тексты с формулами, но для его нормальной работы требуется весьма мощный компьютер, который доступен не каждому. Более того, если вы подготовили текст с формулами в одной из названных систем и решили послать его в виде файла за рубеж (коллеге или в редакцию), то можете с интересом обнаружить, что файлы для ChiWriter или Word for Windows можно обработать далеко не на всех компьютерах (на Западе, в отличие от нашей страны, IBM PC отнюдь не является стандартом de facto). Что же делать?

Оказывается, существует программа, лишенная всех перечисленных недостатков:

- При печати получается текст типографского качества.
- В текст можно включать сколь угодно сложные математические формулы, которые прекрасно смотрятся на печати.

- Подготовленный в виде файла текст не привязан к какому-то одному типу компьютеров, поскольку эта программа реализована практически на всех существующих в настоящее время платформах и действительно работает на них одинаково.
- Реализация этой программы для IBM PC распространяется свободно, так что отечественный пользователь имеет заодно и отличную возможность поработать с легальным программным обеспечением.

Эта замечательная программа называется \TeX (произносится «тэх»). Она создана выдающимся американским математиком и программистом Дональдом Кнутом в конце 70-х годов; издательские системы на ее базе, обладающие всеми перечисленными выше достоинствами, по сию пору широко используются и сдавать позиции не собираются — редкое в компьютерном мире долголетие!

Настоящее пособие посвящено популярной издательской системе, основанной на $\text{\TeX}'e$, — системе \LaTeX . Оно пригодится как читателю, которому необходимо по роду своей работы готовить тексты с формулами, так и специалисту по компьютерной верстке. Никаких познаний в программировании или полиграфии для чтения большей части книги не требуется: достаточно минимальный опыт работы на персональном компьютере в качестве пользователя.

Второе издание дополнено материалом, относящимся к возможностям и особенностям следующей версии $\text{\LaTeX}'a$ — системы $\text{\LaTeX} 2\varepsilon$.

Выражаю глубокую благодарность А. Шеню, без многочисленных бесед и споров с которым эта книга никогда не была бы написана (собственно говоря, и вся эта книга возникла из попытки дополнить книгу [5], представляющую собой пересказ с немецкого языка пособия [4] с дополнениями). Я рад возможности поблагодарить редактора книги И. А. Маховую за высококвалифицированное редактирование. Выражаю также глубокую благодарность В. Д. Арнольду за большую дружескую поддержку.

Глава I

Элементарное введение

1. Общие замечания

1.1. Что такое TeX и L^AT_EX

TeX (произносится «тэх», пишется также «TeX») — это созданная замечательным американским математиком и программистом Дональдом Кнутом (Donald E. Knuth) система для верстки текстов с формулами. Сам по себе TeX представляет собой специализированный язык программирования (Кнут не только придумал язык, но и написал для него транслятор, причем таким образом, что он работает совершенно одинаково на самых разных компьютерах), на котором пишутся издательские системы, используемые на практике. Точнее говоря, каждая издательская система на базе TeX'a представляет собой пакет макропределений (макропакет) этого языка. L^AT_EX (произносится «латех» или «лайтэх», пишется также «LaTeX») — это созданная Лесли Лэмпортом (Leslie Lamport) издательская система на базе TeX'a.

Прежде чем углубиться в изучение собственно L^AT_EX'a, скажем несколько слов о других издательских системах на базе TeX'a. Наряду с L^AT_EX'ом, распространены также макропакеты Plain TeX и A_MS-TeX. Макропакет Plain TeX был разработан самим Дональдом Кнутом, рассматривавшим его в качестве платформы для построения более сложных систем; на практике он используется и как средство для обмена текстами (текст, подготовленный для Plain TeX'a, сравнительно несложно переделать в исходный текст для того же L^AT_EX'a). Что касается A_MS-TeX'a, то эта издательская система ориентирована на важный, но узкий круг приложений: верстку статей для математических журналов и книг, издаваемых Американским Математическим Обществом. Соответственно, в A_MS-TeX'e предусмотрено большое количество весьма изощренных возможностей для создания сложных математических

формул, но при этом нет многих вещей, которые естественно было бы ожидать в издательских системах общего назначения (например, автоматической нумерации частей документа). *LATEX* в этом отношении более гармоничен. Вслед за *AMS-TEX*'ом появился *AMS-LATEX*, призванный сочетать мощь *LATEX*'а как издательской системы и изощренные возможности набора формул, предоставляемые *AMS-TEX*'ом.

Предыдущее издание этой книги было посвящено описанию *LATEX*'а версии 2.09. Эта версия, вышедшая в 1989 году, завершает серию усовершенствований и исправлений в *LATEX*'е, начавшуюся с момента его создания в 1984 году, и до сих пор она была общепринятой. Это положение, однако, не будет длиться долго: в 1994 году вышла принципиально новая версия *LATEX*'а, называемая *LATEX 2 ϵ* (ее создатели — F. Mittelbach, R. Schöpf, M. Goossens и A. Самарин). По сравнению с «классическим» *LATEX*'ом 2.09 *LATEX 2 ϵ* обладает многими новыми привлекательными возможностями (в частности, в качестве составной части он включил в себя и *AMS-LATEX*). Тем не менее на настоящий момент (лето 1995 года) *LATEX 2 ϵ* не вытеснил «старый» *LATEX* даже на Западе; тем более не получил он пока должного распространения в России. В связи с этим наше изложение строится следующим образом. Все, что сказано про *LATEX* вообще, без указания версии, относится в равной мере и к «старому» *LATEX*'у версии 2.09, и к *LATEX*'у 2 ϵ (80% основного текста книги именно таковы).

Если в тексте встречается такой фрагмент, это значит, что он
специально посвящен особенностям *LATEX*'а 2 ϵ .

2.09 А вот так обозначаются фрагменты текста, относящиеся ис-
2.09 ключительно к *LATEX*'у версии 2.09.

Некоторые разделы книги посвящены только *LATEX*'у 2 ϵ или только *LATEX*'у 2.09; это всегда отражается в заглавии раздела (например, «Набор коммутативных диаграмм в *LATEX*'е 2 ϵ » или «Смена шрифтов в тексте: *LATEX* 2.09»). Наконец, мелким шрифтом набран текст, который при первом чтении можно пропустить.

1.2. Достоинства и недостатки

Все издательские системы на базе *TeX*'а обладают достоинствами, заложенными в самом *TeX*'е. Для новичка их можно описать одной фразой: напечатанный текст выглядит «совсем как в книге». *LATEX*, как издательская система, предоставляет удобные и гибкие средства достичь этого полиграфического качества. В частности, указав с помощью простых средств логическую структуру текста, автор может не

вникать в детали оформления, причем эти детали при необходимости нетрудно изменить (чтобы, скажем, сменить шрифт, которым печатаются заголовки, не надо шарить по всему тексту, меняя все заголовки, а достаточно заменить одну строчку в «стилевом файле»). Такие вещи, как нумерация разделов, ссылки, оглавление и т. п. получаются почти что «сами собой».

Огромным достоинством систем на базе TeX'a является высокое качество и гибкость верстки абзацев и математических формул (в этом последнем отношении TeX до сих пор не превзойден).

TeX (и все издательские системы на его базе) неприхотлив к технике: им вполне можно пользоваться, например, на компьютерах на базе 80286-процессора¹, а исходные тексты можно готовить и на совсем уж примитивных машинах.

• Для нормальной работы с LATEX'ом 2_E мощности 286 компьютера • все же недостаточно.

С другой стороны, TeX'овские файлы обладают высокой степенью переносимости: вы можете подготовить LATEX'овский исходный текст на своей IBM PC, переслать его (скажем, по электронной почте) в издательство, и быть уверенным, что там ваш текст будет правильно обработан и на печати получится в точности то же, что получилось у вас при пробной печати на вашем любимом матричном принтере (с той единственной разницей, что фотонаборный автомат даст текст более высокого качества). Благодаря этому обстоятельству TeX стал очень популярен как язык международного обмена статьями по математике и физике. В настоящий момент существует более полутора десятков «электронных журналов» по различным разделам математики и физики, статьи в которых хранятся в виде файлов на компьютерах и рассылаются подписчикам по электронной почте; все эти статьи набраны их авторами в TeX'e.

Есть у TeX'a и недостатки. Первый из них (разделяемый TeX'ом со всеми другими издательскими системами) таков: он работает относительно медленно, занимает много памяти, а полученный результат трудно напечатать на дешевом принтере. Вторая особенность TeX'a, которая может не понравиться тем, кто привык к редакторам наподобие ChiWriter — это то, что он не является системой типа WYSIWYG: работа с исходным текстом и просмотр того, как текст будет выглядеть на печати, — разные операции. Впрочем, благодаря этой особенности время на подготовку текста типографского качества существенно сокращается.

¹ Есть даже реализации TeX'a на IBM XT, но при работе на таких компьютерах производительность труда будет слишком низкой.

Далее, хотя параметры оформления менять легко, создать принципиально новое оформление (новый «стиль») — непростое дело. Впрочем, *LATEX* предоставляет довольно широкие возможности для модификации стандартных стилей.

Наконец, переносимость *TeX*'овских текстов снижается, если в них предусмотрен импорт графических файлов (эта возможность в *TeX*'е зависит от его реализации).

1.3. Литература по *TeX*'у

Каноническое описание языка *TeX* и макропакета Plain *TeX* — серьезная книга Дональда Кнута [3]. У рядового пользователя *LATEX*'а необходимость читать эту книгу обычно не возникает.

Каноническое описание *AMS-TeX*'а — книга [6], также написанная самим создателем *AMS-TeX*а Майклом Спиваком (Michael Spivak).

Наконец, каноническое описание *LATEX*'а — книга Лесли Лэмпорта [1]. К сожалению, в ней недостаточно освещен такой важный на практике вопрос, как модификация стандартных *LATEX*'овских стилей. Настоящее пособие в чем-то уже, чем книга [1], а в чем-то — шире: мы не упоминаем о некоторых экзотических средствах *LATEX*'а, которые либо не слишком полезны на практике, либо дублируют аналогичные средства Plain *TeX*'а, но при этом рассказываем о многих полезных вещах, о которых в [1] не упоминается.

- ε Второе издание книги [1] ориентировано уже на *LATEX* 2_ε (см. [2]). Весьма полное описание новых возможностей, предоставляемых *LATEX*'ом 2_ε, содержится в книге [7]. Имейте, впрочем, в виду, что эта книга уже кое в чем отстала от жизни, поскольку *LATEX* 2_ε продолжал развиваться и совершенствоваться (в том числе и усилиями некоторых ее авторов) и после сдачи ее в печать. Еще одно пособие по *LATEX*'у 2_ε — книга [9].

Первой книгой по *LATEX*'у на русском языке была брошюра [5], представляющая собой выполненный А. Шенем перевод краткого руководства [4] вместе с дополнением переводчика, посвященным описанию одной из популярных реализаций *TeX*'а на IBM PC — системы em*TeX*.

1.4. Как проходит работа с системой *LATEX*

В дальнейшем мы будем отмечать, какие свойства системы специфичны для *LATEX*'а, а какие относятся вообще к *TeX*'у и ко всем издательским системам на его базе, но при первом чтении вы можете об этих тонкостях не задумываться и воспринимать слова *TeX* и *LATEX* как синонимы.

В частности, все, что сказано в этом разделе, применимо не только к $\text{\LaTeX}'y$, но и к любому другому макропакету для $\text{\TeX}'a$, хотя мы для краткости будем говорить только про \LaTeX .

Для начала автор должен подготовить, с помощью любого текстового редактора, файл с текстом, оснащенным командами для $\text{\TeX}'a$, который по традиции имеет расширение `tex` (описанию того, как именно его готовить, и посвящена вся эта книга). Дальнейшая работа протекает в два этапа. Сначала надо обработать файл с помощью программы-транслятора; в результате получается файл с расширением `dvi` (device-independent — не зависящий от устройства).

Теперь полученный файл (его называют еще `dvi`-файлом) можно, с помощью программ, называемых `dvi`-драйверами, распечатать на лазерном или точечно-матричном принтере, посмотреть на экране (текст будет в таком же виде, как он появится на печати) и т. д. (для разных устройств есть разные драйверы). Неудовлетворенный результатом, автор вносит изменения в исходный файл — и цикл повторяется.

На самом деле повторений цикла будет больше, так как при синтаксических ошибках в исходном тексте транслятор будет выдавать сообщения об ошибках, которые приходится исправлять.

Перед тем, как начать работать в системе \LaTeX , вам необходимо четко уяснить для себя три вопроса:

- Что нужно сделать, чтобы оттранслировать исходный текст (т. е. создать из него `dvi`-файл)?
- Что нужно сделать, чтобы просмотреть `dvi`-файл на экране?
- Что нужно сделать, чтобы напечатать `dvi`-файл?

Кроме того, для создания исходного текста нужно, естественно, уметь обращаться с каким-нибудь текстовым редактором.

2. Основные понятия

2.1. Исходный файл

Исходный файл для системы \LaTeX представляет собой собственно текст документа вместе со специальными и командами, с помощью которых системе передаются указания касательно размещения текста. Этот файл можно создать любым текстовым редактором, но при этом необходимо, чтобы в итоге получился так называемый «чистый» текстовый файл (`ASCII`-файл). Это означает, что текст не должен содержать шрифтовых выделений, разбивки на страницы и т. п.

Исходный текст документа *не должен* содержать переносов (*TeX* *сделает их сам*). Слова отделяются друг от друга пробелами, при этом *TeX* не различает, сколько именно пробелов вы оставили между словами (чтобы вручную управлять пробелами между словами, есть специальные команды, о которых пойдет речь позже). Конец строки также воспринимается как пробел. Отдельные абзацы должны быть отделены друг от друга пустыми строками (опять-таки все равно, сколько именно пустых строк стоит между абзацами, важно, что была хоть одна).

В правой колонке приведен фрагмент исходного текста, а в левой — то, как он будет выглядеть на печати после обработки системой *LATeX*.

Слова разделяются пробелами, а
абзацы — пустыми строками.

Абзацный отступ в исходном
тексте оставлять не надо: он по-
лучается автоматически.

Слова разделяются пробелами,
а абзацы ---
 пустыми строками.

Абзацный отступ в исходном
тексте оставлять
не
надо: он получается
автоматически.

Как мог заметить читатель из этого примера, тратить время на фор-
матирование исходного текста тоже незачем.

2.2. Спецсимволы

Большинство символов в исходном тексте прямо обозначает то, что будет напечатано (если в исходном тексте стоит запятая, то и на печати выйдет запятая). Следующие 10 символов:

{ } \$ & # % _ ^ ~ \

имеют особый статус; если вы употребите их в тексте «просто так», то скорее всего получите сообщение об ошибке (и на печати не увидите того, что хотелось). Печатное изображение знаков, соответствующих первым семи из них, можно получить, если в исходном тексте поставить перед соответствующим символом без пробела знак \ (по-английски он называется «backslash»):

Курс тугрика повысился на 7%, и
теперь за него дают \$200.

Курс тугрика повысился на
7\%, и теперь за него
даают \\$200.

Если символ `%` употреблен в тексте не в составе комбинации `\%`, то он является «символом комментария»: все символы, расположенные на строке после него, TeX игнорирует (в том числе и сам `%`). С помощью символа `%` в исходный текст можно вносить пометки «для себя»:

Это пример.

Жил-был у бабушки серенький
козлик.

Это `%` глупый

`%` Лучше: поучительный <-----
пример.

Жил-был у бабушки
серенький козлик.

Обратите внимание на предпоследнюю строку: после знака процента игнорируется вся строка, включая ее конец, который в нормальных условиях играет роль пробела; с другой стороны, начальные пробелы в строке игнорируются всегда. Поэтому TeX не видит пробела между кусками слов `серен` и `ий`, и они благополучно складываются в слово «серенький».

Скажем вкратце о смысле остальных спецсимволов. Фигурные скобки ограничивают группы в исходном файле (см. с. 20). Знак доллара ограничивает математические формулы (см. гл. II). При наборе математических же формул используются знаки `_` и `^` («знак подчеркивания» и «крышка»). Знак `-` обозначает «неразрывный пробел» между словами (см. с. 89). Со знака `\` начинаются все TeX'овские команды (см. разд. 2.3). Знаки `#` и `&` используются в более сложных конструкциях TeX'a, о которых сейчас говорить преждевременно.

Наконец, символы

`<` `>` `|`

в тексте употреблять можно в том смысле, что сообщения об ошибке это не вызовет, но напечатается при этом нечто, совсем на эти символы не похожее. Подлинное место для этих символов, так же как и для символов `=` и `+` — математические формулы, о которых речь пойдет попозже.

2.3. Команды и их задание в тексте

Задание печатного знака процента с помощью последовательности символов `\%` — пример важнейшего понятия TeX'a, называемого *командой*. С точки зрения их записи в исходном тексте, команды делятся на два типа. Первый тип — команды, состоящие из знака `\` и одного символа после него, не являющегося буквой. Именно к этому типу относятся команды `\{`, `\}`, ... `\%`, о которых шла речь на с. 16.

Команды второго типа состоят из \ и последовательности букв, называемой *именем команды* (имя может состоять и из одной буквы). Например, команды `\TeX` и `\LaTeX` генерируют эмблемы систем `\TeX` и `\LaTeX` соответственно. В имени команды, а также между \ и именем, не должно быть пробелов; имя команды нельзя разрывать при переносе на другую строку.

- Команда `\LaTeXe` генерирует эмблему `\LaTeX 2\epsilon`. Разумеется, в
- `\LaTeXe` 2.09 ей пользоваться нельзя.

В именах команд прописные и строчные буквы различаются. Например, `\large`, `\Large` и `\LARGE` — это три разные команды (как вы в дальнейшем узнаете, они задают различные размеры шрифта).

После команды первого типа (из \ и не-буквы) пробел в исходном тексте ставится или не ставится в зависимости от того, что вы хотите получить на печати:

В чем разница между \$1 и \$ 1?

В чем разница между
\\$1 и \\$ 1?

После команды из \ и букв в исходном тексте *обязательно* должен стоять либо пробел, либо символ, не являющийся буквой (это необходимо, чтобы `\TeX` смог определить, где кончается имя команды и начинается дальнейший текст). Вот примеры с командой `\sl` (она переключает шрифт на наклонный):

38 попугаев.

`\sl`38 попугаев.

Подарок мартышке.

`\sl` Подарок мартышке.

Если бы мы написали `\slПодарок мартышке`, то при трансляции `\TeX` зафиксировал бы ошибку (типичную для начинающих) и выдал сообщение о том, что команда `\slПодарок` не определена.

С другой стороны, если после команды из \ и букв в исходном тексте следуют пробелы, то при трансляции они игнорируются. Если необходимо, чтобы `\TeX` все-таки «увидел» пробел после команды в исходном тексте (например, чтобы сгенерированное с помощью команды слово не сливалось с последующим текстом), надо этот пробел специально организовать. Один из возможных способов — поставить после команды пару из открывающей и закрывающей фигурных скобок {} (так что `\TeX` будет знать, что имя команды кончилось), и уже после них сделать пробел, если нужно. Иногда можно также поставить команду \ (backslash с пробелом после него), генерирующую пробел. Вот пример.

Освоить \LaTeX проще, чем \TeX . Человека, который знает систему \TeX и любит ее, можно назвать \TeXником .

Освоить \LaTeX проще, чем \TeX . Человека, который знает систему \TeX и любит ее, можно назвать \TeX ником .

В последней строке этого примера мы сознательно не создали пробела после команды \TeX , чтобы эмблема $\text{\TeX}'$ а слилась с последующим текстом.

2.4. Структура исходного текста

\LaTeX -файл должен начинаться с команды

`\documentstyle`

задающей стиль оформления документа. Пример:

`\documentstyle{book}`

Слово `book` в фигурных скобках указывает, что документ будет оформлен, как книга: все главы будут начинаться с нечетной страницы, текст будет снабжен колонтитулами некоторого определенного вида и т. п. Кроме стиля `book`, в стандартный комплект $\text{\LaTeX}'$ а входят стили `article` (для оформления статей), `report` (нечто среднее между `article` и `book`) и `letter` (для оформления деловых писем так, как это принято в США). Чтобы задать оформление документа с помощью одного из этих стилей, надо в фигурных скобках после команды `\documentstyle` указать вместо `book` название требуемого стиля. Стандартные стили можно (а иногда и нужно) менять, можно создавать и новые стили, но пока что будем исходить из стандартных стилей.

После команды `\documentstyle` могут следовать команды, относящиеся ко всему документу и устанавливающие различные параметры оформления текста, например, величину абзацного отступа (вообще-то все эти параметры определяются используемым стилем, но может случиться, что вам понадобится сделать в них изменения). Далее должна идти команда

`\begin{document}`

Только после этой команды может идти собственно текст. Если вы поместите текст или какую-нибудь команду, генерирующую текст (например, \LaTeX) до `\begin{document}`, то \LaTeX выдаст сообщение об ошибке. Часть файла, расположенная между командами `\documentstyle` и `\begin{document}`, называется преамбулой.

Заканчиваться файл должен командой

```
\end{document}
```

Если даже после `\end{document}` в файле и написано еще что-то, L^AT_EX это проигнорирует.

Следующий пример показывает минимальный L^AT_EX-файл, составленный по всем правилам. Ничего интересного в результате его обработки не напечатается, но уж зато и сообщения об ошибках вы не получите.

```
\documentstyle{article}
\begin{document}
Проба пера.
\end{document}
```

- ε В L^AT_EX'е 2_ε вместо команды `\documentstyle` надо использовать команду `\documentclass`. То, что в L^AT_EX'е 2.09 называлось «стилем документа», называется **классом**. Кроме классов `article`, `report`, `book` и `letter`, L^AT_EX 2_ε поддерживает и некоторые другие. Например, существует класс `amsart`, воспроизводящий все возможности A_MS^TE_X'а. Речь об этом пойдет позже.

Если передать для обработки L^AT_EX'у 2_ε файл, написанный для L^AT_EX'а 2.09 (т. е. начинающийся с команды `\documentstyle`), то L^AT_EX 2_ε перейдет в «режим совместимости» и будет (более или

- ε менее) имитировать работу L^AT_EX'а 2.09.

2.5. Группы

Вторым важнейшим понятием T_EX'а является понятие **группы**. Чтобы понять, что это такое, рассмотрим пример.

При обработке T_EX'ом исходного файла набор текста в каждый момент идет каким-то вполне определенным шрифтом (он называется текущим шрифтом). Изначально текущим шрифтом является «обычный» прямой шрифт (по-английски «roman»). Команда `\sl`, с которой мы уже столкнулись в разд. 2.3, переключает текущий шрифт на наклонный, а команда `\bf` — на полужирный:

Полужирный шрифт начнется со следующего слова. Теперь наклонный шрифт, и снова полужирный, до нового переключения.

Полужирный шрифт начнется со `\bf` следующего слова. Теперь `\sl` наклонный шрифт, `\bf` и снова полужирный, до нового переключения.

Но как же быть, если нужно печатать полужирным шрифтом не весь текст, а только его часть? Можно было бы включить в текст команду `\textbf{}`, переключающую шрифт снова на «обычный». Но есть более простой способ: часть текста, которую вы хотите оформить полужирным шрифтом, можно заключить в фигурные скобки, и дать команду `\bf` *внутри* этих скобок! Тогда сразу же после закрывающей фигурной скобки `\TeX` «забудет» про то, что шрифт переключался на полужирный, и будет продолжать набор тем шрифтом, который был до скобок:

Полужирным шрифтом набрано
только **это слово**; после скобок все
идет, как прежде.

Полужирным шрифтом набрано
только **{\bf это}**
слово; после скобок все
идет, как прежде.

Сами по себе фигурные скобки не генерируют никакого текста и не влияют на шрифт; единственное, что они делают — это ограничивают *группу* внутри файла. Как правило, задаваемые командами `\TeX'`а изменения различных параметров (в нашем случае — текущего шрифта) действуют в пределах той группы, внутри которой была дана соответствующая команда; по окончании группы (после закрывающей фигурной скобки, соответствующей той фигурной скобке, что открывала группу) все эти изменения забываются и восстанавливается тот режим, который был до начала группы. Проиллюстрируем все сказанное следующим примером, в котором используется еще команда `\it` (она переключает шрифт на курсивный):

Сначала переключим шрифт на полужирный, затем на курсив; временно перейдем снова на полужирный; посмотрите, как восстановится шрифт после конца группы.

Сначала {переключим
шрифт `\bf` на полужирный,
затем на `\it` курсив;
временно перейдем
снова на `{\bf полужирный;}`
посмотрите, как
восстановится} шрифт
после конца группы.

Как видите, группы могут быть вложены друг в друга. Обратите внимание, что внутри внешней группы полужирный шрифт начался не с того места, где была открывающая скобка, а только после команды `\bf` (именно команда, а не скобка, переключает шрифт). Шутки ради мы создали еще одну группу из двух последних буквы слова конца, первой буквы слова группы и пробела между ними; как и должно быть, на печати это никак не отразилось: ведь внутри скобок мы ничего не делали!

Трюк с постановкой пары скобок {} после имени команды, о котором шла речь на с. 18 — тоже пример использования групп. В этом случае скобки ограничивают «пустую» группу; ставятся они в качестве не-букв, ограничивающих имя команды и при этом никак не влияющих на печатный текст.

Фигурные скобки в исходном тексте должны быть сбалансированы²: каждой открывающей скобке должна соответствовать закрывающая. Если вы почему-либо не соблюли это условие, при трансляции вы получите сообщение об ошибке.

Некоторые команды, называемые *глобальными*, сохраняют свое действие и за пределами той группы, где они были употреблены. Всякий раз, когда идет речь о глобальной команде, это будет специально оговариваться.

2.6. Параметры

Наряду с текущим шрифтом, о котором уже шла речь, \TeX в каждый момент обработки исходного текста учитывает значения различных параметров, таких, как величина абзацного отступа, ширина и высота страницы, расстояние по вертикали между соседними абзацами, а также великое множество других важных вещей. Расскажем, как можно менять эти параметры, если это понадобится.

Параметры \TeX 'а обозначаются аналогично командам: с помощью символа \ («backslash»), за которым следует либо последовательность букв, либо одна не-буква. Например, \parindent обозначает в \TeX 'е величину абзацного отступа; если нам понадобилось, чтобы абзацный отступ равнялся двум сантиметрам, можно написать так:

```
\parindent=2cm
```

Аналогично поступают и в других случаях: чтобы изменить параметр, надо написать его обозначение, а затем, после знака равенства, значение, которое мы «присваиваем» этому параметру; в зависимости от того, что это за параметр, это может быть просто целое число, или длина (как в разобранном примере), или еще что-нибудь.

2.7. Команды с аргументами

Команды наподобие \LaTeX или, скажем, \bf действуют «сами по себе»; многим командам, однако, необходимо задать *аргументы*. Первый пример тому дает команда \documentstyle (а в \LaTeX 'е 2ε —

²Это не относится к скобкам, входящим в состав команд \{ и \}.

`\documentclass`): слово, указываемое в фигурных скобках — ее аргумент; если его не указать, то произойдет ошибка. В \LaTeX 'е аргументы команд бывают обязательные и необязательные. Обязательные аргументы задаются в *фигурных скобках*; если для команды предусмотрено наличие обязательных аргументов, она без них правильно работать не будет. У многих команд предусмотрены также и необязательные аргументы, которые влияют на работу команды, коль скоро они указаны, но при этом нормальная работа команды не нарушится и при их отсутствии. Необязательные аргументы задаются в *квадратных скобках*.

В частности, у команды `\documentstyle` (`\documentclass`) предусмотрен один обязательный аргумент, о котором уже шла речь, и один необязательный: в квадратных скобках перед обязательным аргументом можно указать список (через запятую) так называемых стилевых опций, т. е. дополнительных особенностей оформления. Например, если мы хотим, чтобы книга набиралась шрифтом кегля 12 вместо кегля 10, принятого по умолчанию³, и притом в две колонки, мы должны начать файл командой

```
\documentstyle[12pt,twocolumn]{book}
```

Есть также стилевая опция `11pt`, означающая, что текст будет набираться кеглем 11. Полный список возможных стилевых опций приведен в разд. 1 главы IV. То, что было выше сказано про команду `\documentstyle`, верно и в $\text{\LaTeX}_2\epsilon$, только вместо `\documentstyle` надо писать `\documentclass`.

- ε Наряду со стилевыми опциями в $\text{\LaTeX}_2\epsilon$ используются и так называемые стилевые пакеты. После начинающей файл команды `\documentclass` может следовать одна или несколько команд `\usepackage`, в аргументе которой стоит, через запятую, список подключаемых этой командой стилевых пакетов. Например, первые две строки файла могут быть:

```
\documentclass[12pt,twocolumn]{book}
\usepackage{amsfonts,longtable}
```

Здесь пакет `amsfonts` подключается, чтобы использовать в математических формулах дополнительные шрифты, позволяющие напечатать что-нибудь вроде $\mathfrak{sl}_2(\mathbb{C})$, а пакет `longtable` нужен, чтобы иметь возможность набирать таблицы, простирающиеся на

³Примечание для полиграфистов: в \TeX 'овских шрифтах кегль исчисляется в англо-американских пунктах, отличающихся от принятых в нашей стране (см. с. 25); \TeX 'овский кегль 10 примерно соответствует нашему девятому кеглю.

несколько страниц. Для $\text{\LaTeX}'a 2\varepsilon$ создано уже весьма много различных стилевых пакетов. Когда мы будем рассказывать о возможностях $\text{\LaTeX}'a 2\varepsilon$, мы часто будем говорить: «чтобы воспользоваться этой возможностью, необходимо подключить стилевой пакет такой-то». В основном тексте книги мы молчаливо предполагаем, что поставка $\text{\LaTeX}'a 2\varepsilon$, которой вы пользуетесь, все эти пакеты содержит. В приложении Ж мы расскажем о том, откуда их взять.

Необязательных аргументов может быть предусмотрено несколько; иногда они должны располагаться до обязательных, иногда после. В любом случае порядок, в котором должны идти аргументы команды, надо строго соблюдать. Между скобками, в которые заключены обязательные аргументы, могут быть пробелы, но не должно быть пустых строк.

2.8. Окружения

Еще одна важная конструкция $\text{\LaTeX}'a$ — это окружение (environment).

Окружение — это фрагмент файла, который начинается с текста

`\begin{Имя_окружения}`

где `{Имя_окружения}` представляет собой первый обязательный (и, возможно, не единственный) аргумент команды `\begin{...}`. Заканчивается окружение командой

`\end{Имя_окружения}`

(команда `\end` имеет только один аргумент — имя завершаемого ею окружения). Например:

`\begin{center}`

Все строки этого абзаца будут центрированы; переносов слов не будет, если только слово, как дезоксирибонуклеиновая кислота, не длинней строки.
`\end{center}`

Все строки этого абзаца будут центрированы; переносов слов не будет, если только слово, как дезоксирибонуклеиновая кислота, не длинней строки.

Таблица I.1. ТЕХ'овские единицы длины

pt	пункт ≈ 0.35 миллиметра
pc	пика = 12pt
mm	миллиметр
cm	сантиметр = 10 mm
in	дюйм = 25,4 mm
dd	пункт Дидо $\approx 1,07$ pt
cc	цицеро = 12 dd

Каждой команде `\begin`, открывающей окружение, должна соответствовать закрывающая его команда `\end` (разумеется, с тем же именем окружения в качестве аргумента).

Важнейшим свойством окружений является то, что они действуют и как фигурные скобки: *часть файла, находящаяся внутри окружения, образует группу*. Например, внутри окружения `center` в вышеприведенном примере можно было бы сменить шрифт, скажем, командой `\it`, и при этом после команды `\end{center}` восстановился бы тот шрифт, что был перед окружением.

2.9. Звездочка после имени команды

В L^AT_EX'е некоторые команды и окружения имеют варианты, в которых непосредственно после имени команды или окружения ставится звездочка *. Например, команда `\section` означает «начать новый раздел документа», а команда `section*` означает «начать новый раздел документа, не нумеруя его». После имени команды со звездочкой пробелы не игнорируются; если команда со звездочкой имеет аргументы, пробела между звездочкой и аргументами быть не должно.

2.10. Единицы длины

Многие параметры, используемые ТЕХ'ом, являются размерами (пример тому мы видели в разд. 2.6); в табл. I.1 собраны единицы длины (кроме нескольких экзотических), которые можно использовать в ТЕХ'е при задании размеров.

Замечание для полиграфистов: ТЕХ'овский пункт является единицей измерения, принятой в англо-американской типометрии; он отличается от пункта, принятого в континентальной Европе (в том числе и в России). Единица измерения, называемая в ТЕХ'е пунктом Дидо, соответствует пункту, к которому привыкли отечественные полиграфисты.

Можно задавать размеры с помощью любой из этих единиц; при записи дробного числа можно использовать как десятичную запятую, так и десятичную точку (в таблице мы использовали оба способа); прописные и строчные буквы в обозначениях единиц длины не различаются.

Даже если длина, которую вы указываете $\text{\TeX}'$ у, равна нулю, все равно необходимо указать при этом нуле какую-нибудь из используемых $\text{\TeX}'$ ом единиц длины. Например, если написать

```
\parindent=0
```

то вы получите сообщение об ошибке; вместо 0 надо было бы писать, например, `Opt` или `0in`.

Кроме перечисленных, в $\text{\TeX}'$ е используются еще две «относительные» единицы длины, размер которых зависит от текущего шрифта. Это `em`, приблизительно равная ширине буквы M текущего шрифта, и `ex`, приблизительно равная высоте буквы x текущего шрифта. Эти единицы удобно использовать в командах, которые должны работать единообразно для шрифтов разных размеров. В частности, расстояние в `1em` на глаз обычно воспринимается как «один пробел».

2.11. Автоматическая генерация ссылок

$\text{L}\text{\TeX}$ предоставляет возможность организовать ссылки на отдельные страницы или разделы документа таким образом, чтобы программа сама определяла номера страниц или разделов в этих ссылках. Объясним это на примере.

Представим себе, что вам нужно сослаться на какое-то место в вашем тексте. Проще всего указать страницу, на которой это место находится, написав «... как мы уже отмечали на с. 99» или что-то в этом роде. Проблема, однако, в том, что заранее нельзя угадать, на какую страницу печатного текста попадет это место. Вместо того чтобы гадать, можно сделать следующее:

- Пометить то место, на которое вы хотите сослаться в дальнейшем (или предшествующем) тексте;
- В том месте текста, где вы хотите поместить ссылку, поставить команду-ссылку на вашу метку.

Конкретно это реализуется так. Помечается любое место текста с помощью команды `\label`. Эта команда имеет один обязательный аргумент (помещаемый, стало быть, в фигурных скобках) — «метку». В качестве метки можно использовать любую последовательность букв, цифр и знаков препинания (не содержащую пробелов, фигурных скобок и символов `-` или `\`). Например, эта команда может иметь вид:

`\label{wash}`

Ссылка на страницу, на которой расположена метка, производится с помощью команды `\pageref`. У нее также один обязательный аргумент — та самая метка, на которую вы хотите сослаться. Пример:

Обязательно мойте руки перед едой, чтобы не заболеть.

Как известно (см. с. 99), руки надо мыть.

Обязательно мойте
руки`\label{wash}` перед
едой, чтобы не заболеть.

Как известно (см.
с. `\pageref{wash}`),
руки надо мыть.

Обратите внимание, что мы поставили команду `\label` рядом с ключевым словом «руки» без пробела, чтобы гарантировать, что будет помечена именно та страница, на которую попало это слово.

В этом примере мы использовали еще значок `,`, чтобы при печати сокращение «с.» попало на ту же строку, что и номер страницы. Подробности см. в разд. III.2.1.

После того как вы впервые вставите в свой файл команды `\label` и `\pageref`, при трансляции вы получите сообщение о том, что ваша ссылка не определена, а на печати или при просмотре увидите на месте своих ссылок вопросительные знаки. Дело в том, что в этот момент `LATEX` еще не знает значения ваших меток: он только записывает информацию о них в специальный файл (с тем же именем, что у обрабатываемого файла, и расширением `aux`); при следующем запуске он прочтет эту информацию и подставит ссылки. В промежутке между двумя запусками в файл могли быть внесены изменения, что может привести к сдвигу нумерации страниц. Если такие изменения действительно произошли, `LATEX` сообщит вам об этом и попросит запустить программу еще раз, чтобы получить корректные ссылки.

Если вы после двух запусков подряд получите сообщение о неопределенной ссылке, значит, в исходном тексте присутствует ошибка (вероятнее всего, опечатка в аргументе команды `\pageref`; возможно, вы забыли включить в текст команду `\label`).

На место, помеченное с помощью команды `\label`, можно сослаться с помощью команды `\ref`, а не `\pageref` — тогда на печати получится не номер страницы, а номер раздела документа, в котором находится метка, или номер рисунка, или номер элемента в «нумерованном перечне»... — пометить с возможностью ссылки можно почти любой элемент документа. Об этом мы будем подробно говорить в гл. IV.

3. Набор формул в простейших случаях

3.1. Основные принципы

В документе, подготовленном с помощью TeX'a, различают математические формулы внутри текста и «выключные» (выделенные в отдельную строку). Формулы внутри текста окружаются знаками \$ (с обеих сторон). Выключные формулы окружаются парами знаков доллара \$\$ и \$\$ с обеих сторон. Формулами считаются как целые формулы, так и отдельные цифры или буквы, в том числе греческие, а также верхние и нижние индексы и спецзнаки. Пробелы внутри исходного текста, задающего формулу, игнорируются (на печати TeX сам сделает нужные пробелы; надо по-прежнему ставить пробелы, обозначающие конец команды); пустые строки не разрешаются. TeX расставляет пробелы в математических формулах автоматически (например, знак равенства окружается небольшими пробелами). Если надо оставить пробел перед или после внутритечстовой формулы, надо оставить его перед или после ограничивающего ее знака доллара. То же самое относится и к знакам препинания, следующим за внутритечстовой формулой: их также надо ставить после закрывающего формулу знака доллара. Каждая буква в формуле рассматривается как имя переменной и набирается шрифтом «математический курсив» (в отличие от обычного курсива, в нем увеличены расстояния между соседними буквами).

Часть файла, составляющая математическую формулу, образует группу (см. с. 20): изменения параметров, произведенные внутри формулы, забываются по ее окончании.

3.2. Степени и индексы

Степени и индексы набираются с помощью знаков ^ и _ соответственно.

Катеты a , b треугольника связаны с гипотенузой с формулой $c^2 = a^2 + b^2$ (теорема Пифагора).

Катеты \$a\$, \$b\$ треугольника связаны с гипотенузой \$c\$ формулой \$c^2=a^2+b^2\$ (теорема Пифагора).

Если индекс или показатель степени — выражение, состоящее более чем из одного символа, то его надо взять в фигурные скобки:

Из теоремы Ферма следует, что уравнение

$$x^{1993} + y^{1993} = z^{1993}$$

не имеет решений в натуральных числах.

Из теоремы Ферма следует, что уравнение

\$\$

$$x^{1993} + y^{1993} = z^{1993}$$

\$\$

не имеет решений в натуральных числах.

Если у одной буквы есть как верхние, так и нижние индексы, то можно указать их в произвольном порядке:

Обозначение $R^i_{;kl}$ для тензора кривизны было введено еще Эйнштейном.

Обозначение $R^i_{\{jkl\}}$ для тензора кривизны было введено еще Эйнштейном.

Если же требуется, чтобы верхние и нижние индексы располагались не один под другим, а на разных расстояниях от выражения, к которому они относятся, то нужно TeX немного обмануть, оформив часть индексов как индексы к «пустой формуле» (паре из открывающей и закрывающей скобок):

Можно также написать $R^i_{;kl}$, хотя не всем это нравится.

Можно также написать $$R_j{}^i{}_{kl}$$, хотя не всем это нравится.

Если вы хотите написать формулу, читающуюся как «два в степени икс в кубе», то запись 2^x^3 не даст ничего, кроме сообщения об ошибке; правильно будет $2^{\{x^3\}}$ (на печати это будет выглядеть как 2^{x^3}).

3.3. Дроби

Дроби, обозначаемые косой чертой (так рекомендуется обозначать дроби во внутритекстовых формулах), набираются непосредственно:

Неравенство $x + 1/x \geq 2$ выполнено для всех $x > 0$.

Неравенство $x+1/x \geq 2$ выполнено для всех $x > 0$.

В этом примере мы еще использовали знаки «строгих» неравенств (в TeX'овских формулах они набираются непосредственно, как знаки $>$ и $<$) и нестрогих неравенств (знак «больше или равно» генерируется командой \geq , «меньше или равно» — командой \leq). Между прочим, если вы употребите символы $<$ и $>$ в обычном тексте, вне формул, то вместо знаков «меньше» и «больше» увидите небольшой сюрприз.

Наряду со знаками для нестрогих неравенств, TeX предоставляет большое количество специальных символов для математических формул (греческие буквы также рассматриваются как специальные символы). Все эти символы набираются с помощью специальных команд (не требующих параметров). Списки этих команд вы найдете в таблицах в начале следующей главы.

Если вы используете в формуле десятичные дроби, в которых дробная часть отделена от целой с помощью запятой, то эту запятую следует взять в фигурные скобки (в противном случае после нее будет оставлен небольшой дополнительный пробел, что нежелательно):

$$\pi \approx 3,14$$

`$\pi\approx 3{,}14$`

Здесь команда `\pi` порождает греческую букву π , а команда `\approx` — знак \approx («приближенно равно»).

Дроби, в которых числитель расположен над знаменателем, набираются с помощью команды `\frac`. Эта команда имеет два обязательных аргумента: первый — числитель, второй — знаменатель. Пример:

$$\frac{(a+b)^2}{4} + \frac{(a-b)^2}{4} = ab$$

`$$`

```
\frac{(a+b)^2}{4}+
\frac{(a-b)^2}{4}=
ab
```

`$$`

Если числитель и/или знаменатель дроби записывается одной буквой (в том числе греческой) или цифрой, то можно их и не брать в фигурные скобки:

$$\frac{1}{2} + \frac{x}{2} = \frac{1+x}{2}$$

`$$\frac{1}{2}+\frac{x}{2}=\frac{1+x}{2}$$`

3.4. Скобки

Круглые и квадратные скобки набираются как обычно, для фигурных скобок используются команды `\{` и `\}`, для других также есть специальные команды, например `\langle` («левая угловая скобка» `(`).

Команда `\left` перед открывающей скобкой в совокупности с командой `\right` перед соответствующей ей закрывающей скобкой позволяет автоматически выбрать нужный размер скобки.

$$1 + \left(\frac{1}{1 - x^2} \right)^3$$

`$$`

```
1+\left(\frac{1}{1-x^{2}}\right)^{3}
```

`$$`

Подробнее о скобках, размер которых выбирается автоматически, рассказано в следующей главе (разд. II.2.6).

3.5. Корни

Квадратный корень набирается с помощью команды `\sqrt`, обязательным аргументом которой является подкоренное выражение; корень произвольной степени набирается с помощью той же команды `\sqrt` с необязательным аргументом — показателем корня (необязательный аргумент у этой команды ставится перед обязательным). Пример:

По общепринятыму соглашению,
 $\sqrt[3]{x^3} = x$, но $\sqrt{x^2} = |x|$.

По общепринятыму соглашению,
 $\$ \sqrt[3]{x^3} = x \$$, но
 $\$ \sqrt{x^2} = |x| \$$.

Обратите внимание, что вертикальные черточки, обозначающие знак модуля, набираются непосредственно.

3.6. Штрихи и многоточия

Штрихи в математических формулах обозначаются знаком ' (и не оформляются как верхние индексы):

Согласно формуле Лейбница,

$$(fg)'' = f''g + 2f'g' + fg''.$$

Это похоже на формулу квадрата суммы.

Согласно формуле Лейбница,

$$\begin{aligned} & \$ \\ & (fg)'' = f''g + 2f'g' + fg'' . \end{aligned}$$

\$\$

Это похоже на формулу
квадрата суммы.

Обратите, кстати, внимание на то, что точку мы поставили в конце выключной формулы (если бы мы поставили ее после знаков \$\$, то с нее начался бы абзац, следующий после формулы).

В математических формулах встречаются многоточия; ТЕХ различает многоточие расположенное внизу строки (обозначается `\ldots`), и расположенное по центру строки (оно обозначается `\cdots`). Первое из них используется при перечислениях, второе — когда нужно заменить пропущенные слагаемые или сомножители (такова американская традиция; в России обычно многоточие ставят внизу строки и в этом случае):

В детстве К.-Ф. Гаусс придумал, как быстро найти сумму

$$1 + 2 + \dots + 100 = 5050;$$

это случилось, когда школьный учитель задал классу найти сумму чисел $1, 2, \dots, 100$.

В детстве К.-Ф. Гаусс придумал, как быстро найти сумму

\$\$

$$1+2+\cdots+100=5050;$$

\$\$

это случилось, когда школьный учитель задал классу найти сумму чисел $\$1,2,\ldots,100\$$.

Знак “ после инициалов великого Гаусса мы поставили, чтобы фамилия не могла перенестись на другую строку отдельно от инициалов (см. с. 89). *Л^AT_EX* позволяет использовать команду `\ldots` и в обычном тексте, вне математических формул, для знака многоточия (см. с. 88).

3.7. Функции типа синус

Функции наподобие `sin`, `log` и т. п., имена которых надо набирать прямым шрифтом, набираются с помощью специальных команд (обычно одноименных с обозначениями соответствующих функций). Полный список таких команд приведен в разд. II.1.2.

Нетрудно видеть, что $\log_{1/16} 2 = -1/4$, а $\sin(\pi/6) = 1/2$.

Нетрудно видеть, что
 $\$\\log_{1/16} 2=-1/4\$,$ а
 $\$\\sin(\\pi/6)=1/2\$.$

Заметьте, что основание логарифма задается как нижний индекс.

В стандартный набор команд *TeX*'а не входят команды для функций `tg` и `ctg` (в англоязычных странах эти функции принято обозначать `tan` и `cot` соответственно). Большой беды тут нет, поскольку эти недостающие команды легко определить самому (см. гл. II); возможно, кроме того, что вы получили *Л^AT_EX* вместе с «русифицирующим стилем», в котором необходимые определения команд уже сделаны.

4. Разбиение исходного файла на части

Команды, рассматриваемые в этом разделе, помогают разумно организовать исходный текст.

Часто бывает удобно разбить большой текст на несколько частей, хранящихся в разных файлах. Чтобы можно было объединить их в одно целое, в *TeX*'е предусмотрена команда `\input`. Если в тексте написать

`\input имя_файла,`

то \TeX будет работать так, как если бы вместо строки с командой `\input` стоял текст файла, имя которого вы указали. Можно также написать

```
\input{имя_файла}
```

взяв имя файла в фигурные скобки.

Обычно, когда готовят текст большого объема, то создают небольшой файл, в котором между `\begin{document}` и `\end{document}` размещены строки с командами `\input`, задающими включение файлов, в которых и записана основная часть текста. Например, книгу из четырех глав, записанных в файлах `ch1.tex`, ..., `ch4.tex`, можно организовать в виде файла из девяти строчек (именно его, а не файлы с отдельными главами, надо будет передать для обработки $\text{\TeX}'у$):

```
\documentstyle[11pt]{report}
\frenchspacing
\pagestyle{plain}
\begin{document}
\input ch1.tex
\input ch2.tex
\input ch3.tex
\input ch4.tex
\end{document}
```

Ради реализма мы в этом примере включили в преамбулу парочку команд, которые могли бы там появиться и в реальной ситуации. Первая из них означает, что промежутки между отдельными предложениями должны иметь тот же размер, что и промежутки между словами в предложении, а вторая — что номера страниц будут печататься снизу и при этом колонтитулов не будет. Позже мы рассмотрим эти вещи подробнее.

Каждую команду `\input` следует располагать на отдельной строке, как в вышеприведенном примере. Если расширение файла, являющегося аргументом команды `\input`, не указано, то \TeX по умолчанию считает, что это расширение имеет вид `tex`.

Если в вашем тексте присутствуют команды `\input`, то в процессе трансляции при начале чтения соответствующего файла на экран выводится его имя, чтобы вы понимали, к какому из ваших файлов будут относиться дальнейшие сообщения $\text{\TeX}'а$ (если таковые будут).

Если вы хотите, чтобы \TeX прочитал только часть вашего файла, можно воспользоваться командой `\endinput`. Если она присутствует в файле, читаемом $\text{\TeX}'ом$ с помощью команды `\input`, то файл будет

прочитан только до строчки, в которой написано `\end{input}`, после чего его чтение прекратится.

5. Обработка ошибок

В исходных текстах для TeX'a, которые вы будете готовить, неизбежно будут присутствовать ошибки. В настоящем разделе мы обсудим, как TeX на них реагирует и как вам, в свою очередь, следует реагировать на эти реакции TeX'a.

Все сообщения, которые TeX выдает на экран в процессе трансляции исходного текста, все ваши ответы на эти сообщения, вообще все, что в процессе трансляции появляется на экране, записывается в специальный файл — протокол трансляции. Обычно файл-протокол имеет то же имя, что обрабатываемый TeX'ом файл, и расширение `log`, поэтому на жаргоне протокол трансляции называется `log`-файлом. Когда трансляция завершена, вы можете в спокойной обстановке просмотреть `log`-файл и проанализировать, что произошло.

Часть информации, выдаваемой при трансляции на экран и в `log`-файл, представляет собой предупреждения (например, о нештатных ситуациях при верстке абзаца), при выдаче которых трансляция не прерывается (в разд. III.7 будет подробно рассказано, что означают такие предупреждения). В случае, однако, если TeX натыкается на синтаксическую ошибку в исходном тексте, трансляция приостанавливается, а на экран выдается сообщение об ошибке.

Чтобы понять, что делать с этими сообщениями, давайте проведем эксперимент. Наберите следующий файл `test.tex` из 14 строк, в котором умышленно допущено несколько ошибок (только не сделайте лишних ошибок при наборе):

```
\documentstyle{article}
\begin{document}
По-английски специалист по \TeX'у называется \TeXpert.
Следующая строка будет центрирована:
\begin{center}
Строка в центре.
\end{center}
А теперь попробуем формулы, например, такие,
как  $3^3=27$ . И еще выключную формулу:
$$\frac{25}{36}=\left.\frac{1}{1+\frac{1}{5}}\right.^2
$$
И последняя формула:  $\sqrt{4}=2$ .
```

\end{document}

Если вы проводите этот эксперимент на $\text{\LaTeX}'e 2\varepsilon$, то, естественно, надо написать \documentclass вместо \documentstyle.

Теперь обработайте наш файл `test.tex` с помощью $\text{\TeX}'a$. Вскоре вы увидите на экране вот что:

! Undefined control sequence.

1.3 ...алист по \TeX'у называется \TeXpert

?

Первая строка $\text{\TeX}'$ овского сообщения об ошибке всегда начинается с восклицательного знака, после которого идет краткое указание на характер ошибки (в нашем случае речь идет о том, что обнаружена несуществующая команда). Второй обязательный элемент сообщения об ошибке — строка, начинающаяся с 1., после которой идет номер строки исходного текста с ошибкой (в нашем случае 3). После номера на экран выдается сама эта строка, или та ее часть, которую \TeX успел прочесть к моменту обнаружения ошибки. В нашем случае текст был прочитан до несуществующей команды \TeXpert включительно (эта «команда» получилась потому, что мы забыли оставить пробел, ограничивающий имя команды \TeX — см. с. 18), на которой \TeX и прервал чтение файла. Наконец, третий основной элемент сообщения об ошибке — строка, состоящая из одного вопросительного знака. Этот вопросительный знак представляет собой «приглашение» пользователю: вам теперь предстоит на сообщение об ошибке отреагировать. Рассмотрим возможные реакции.

Во-первых, на худой конец всегда можно нажать клавишу `x` или `X` (латинскую) и после этого «ввод»: тогда трансляция немедленно завершится. Делать так сразу же, однако, не очень разумно: в тексте вполне могут быть и другие ошибки, и за один сеанс хочется обнаружить их побольше. Поэтому лучше просто нажать клавишу «ввод»: при этом \TeX исправит обнаруженную ошибку «по своему разумению» и продолжит трансляцию. Догадаться о том, что ошибка произошла именно из-за забытого пробела, программа, естественно, не может: исправление будет заключаться попросту в том, что будет проигнорирована несуществующая команда \TeXpert (так что из печатного текста будет неясно, как по-английски называют специалиста по $\text{\TeX}'u$). Нажимать «ввод» в ответ на сообщение об ошибке — наиболее распространенная на практике реакция: в 90% случаев этого вполне достаточно, и на первых порах можно этим и ограничиться. Если вы твердо намерены нажимать на «ввод» в ответ на все сообщения об ошибках, то можно в

ответ на первое же из этих сообщений нажать на S или s, а затем на «ввод»; при обнаружении дальнейших ошибок трансляция прерываться не будет (\TeX будет обрабатывать ошибки так, как если бы вы все время нажимали на «ввод»), по экрану пронесутся сообщения об ошибках, а затем вы сможете их спокойно изучить, просмотрев log-файл.

Итак, трансляция продолжается. Следующая остановка будет с таким сообщением:

```

LaTeX error. See LaTeX manual for explanation.
Type H <return> for immediate help.
! \begin{center} ended by \end{centrr}.
\@latexerr ...diate help.\errmessage {#1}

\@checkend ...urrenvir \else \@badend {#1}
                           \fi
\end ...me end#1\endcsname \@checkend {#1}
                           \expandafter \endgroup \if@...
1.7 \end{centrr}

```

?

Это сообщение об ошибке начинается со слов **LaTeX error**. Такого рода сообщения не встроены в \TeX , а создаются \LaTeX'ом (так что в принципе можно создать «русифицирующий стиль» для $\text{\LaTeX}'а$, при пользовании которым эти сообщения будут выдаваться по-русски). В нем, однако, по-прежнему присутствуют три основных элемента сообщений об ошибке: строка, начинающаяся с !, строка, начинающаяся с 1., и приглашение — вопросительный знак (на все остальное в этом сообщении внимания не обращайте — это интересно только \TeX никам). Присутствует на экране и объяснение ошибки: из-за опечатки (`centrr` вместо `center`) получилось, что команда `\begin{center}`, открывающая окружение, не соответствует команде `\end{center}`, закрывающей его (см. разд. 2.8: имена окружений при открывающем `\begin{center}` и закрывающем `\end{center}` должны совпадать). Так или иначе, давайте снова нажмем на «ввод»; тут же мы увидим вот что:

```

! Missing $ inserted.
<inserted text>
               $
<to be read again>

```

1.9 как $(2x+1)^-$

$3=5x\$$. И еще выключную формулу:

На сей раз мы забыли знак доллара, открывающий формулу; \TeX , однако, понял это не сразу, а лишь наткнувшись на символ \wedge , который вне формул таким образом использовать нельзя. Нажмем «ввод»: \TeX исправит положение, вставив знак доллара непосредственно перед знаком \wedge , и пойдет дальше (все такие исправления не вносятся в ваш файл, а происходят только в оперативной памяти компьютера). На печати формула будет иметь странный вид, поскольку $(2x+1)$ будет набрано прямым шрифтом, а $5x$ — курсивным, но зато \TeX сможет продолжить трансляцию (и искать дальнейшие ошибки).

Следующая ошибка будет уже знакомого нам типа, только на сей раз несуществующая команда получается не из-за забытого пробела, а из-за опечатки ($\backslash lrft$ вместо $\backslash left$):

```
! Undefined control sequence.
1.10 $$\backslashfrac{25}{36}=\backslashlrft
                           (\backslashfrac{1}{}
?
```

Нажмем очередной раз на «ввод», и немедленно увидим сообщение еще об одной ошибке:

```
! Extra \right.
1.11   {1+\backslashfrac{1}{5}}\right)
                  ^2
?
```

Откуда такое сообщение, если в строке 11 у нас все правильно? Оказывается, эта ошибка была «наведена» предыдущей! В самом деле, перед этим \TeX проигнорировал «команду» $\backslash lrft$, набранную вместо $\backslash left$ (именно так \TeX и делает, если в ответ на ошибку «несуществующая команда» нажать на клавишу «ввод»), так что команду $\backslash left$ \TeX вообще «не видел»; теперь выходит так, что в тексте, который видит \TeX , присутствует команда $\backslash right$ без команды $\backslash left$, что запрещено (см. разд. II.2.6). Ввиду возможности появления таких «наведенных» ошибок, исправлять ошибки надо, начиная с самой первой; не исключено, что при ее исправлении часть последующих пропадет сама собой.

Нажмем на «ввод» и на этот раз; \TeX опять по-свойски исправит ошибку, и вскоре вы увидите последнее сообщение об ошибке:

```
! Missing } inserted.
<inserted text>
      }
<to be read again>
$
```

1.13 И последняя формула: $\sqrt{4} = 2$

?

На сей раз ошибка в том, что мы забыли закрывающую фигурную скобку. Нажмем на «ввод»; ТЕХ вставит недостающую скобку (в результате чего на печати получится забавная «формула» $\sqrt{4} = 2$, соответствующая исходному тексту $\sqrt{4}=2$: пропажа закрывающей скобки обнаружилась не там, где мы ее забыли, а там, где ее отсутствие вошло в противоречие с синтаксическими правилами ТЕХ'а), после чего трансляция наконец завершится. Кстати, цифра 1 в квадратных скобках, появляющаяся при этом на экране, означает, что ТЕХ сверстал страницу номер 1 и записал ее содержимое в dvi-файл. Теперь можно и просмотреть, как будет выглядеть наш текст на печати; из-за многочисленных ошибок вид будет несколько странный.

Количество различных сообщений об ошибках, которые может выдавать ТЕХ, составляет несколько сотен, и нормальная реакция на них обычно такая же, как в нашем эксперименте, принципиальных отличий вы не увидите (если вы не ТЕХник, конечно); сейчас мы рассмотрим еще две типичные ошибки, реакция на которые должна быть иной.

Во-первых, может случиться, что в качестве аргумента команды \input задано имя несуществующего файла. В этом случае вы получите сообщение наподобие следующего:

```
? I can't find file 'ttst.tex'.
```

```
1.16 \input ttst
```

Please type another input file name:

В ответ на это следует набрать правильное имя файла и нажать на «ввод», и трансляция благополучно продолжится. Простое нажатие на «ввод» тут ничего не даст. Если вообще никакого файла нет (например, ТЕХ запущен по ошибке), наберите null — это всегда существующий пустой файл. (В некоторых версиях — null с одним 1).

- ε Если вы пользуетесь LATEX'ом 2_E, то на эту ошибку можно реагировать почти так же, как и на любую другую: нажать x и «ввод» (трансляция прервется) или z и «ввод» (неправильная команда \input будет проигнорирована, на дальнейшие ошибки ТЕХ будет реагировать так, как если бы вы все время нажимали «ввод»). А если вы имеете счастье работать с LATEX'ом 2_E под операционной системой UNIX, то в ответ на такое сообщение об ошибке можно и попросту нажать «ввод», и она будет проигнорирована.

Если команда `\input` с именем несуществующего файла попадется TeX'у после того, как вы в ответ на какую-то из прежних ошибок сказали `s`, то трансляция на этом месте тем не менее остановится и TeX поинтересуется верным именем файла.

Вторая ошибка, о которой мы хотели сказать, строго говоря ошибкой не является; скорее, это нештатная ситуация. Чтобы смоделировать ее, проведем такой эксперимент: удалим из нашего файла `test.tex` последнюю строчку, гласящую `\end{document}`, и снова запустим LATEX для обработки этого файла. Нажав сколько-то раз «ввод», мы обнаружим, что работа TeX'a не закончилась, а на экран выдана звездочка: `*`. Эта звездочка — приглашение TeX'a ввести еще текст или команды; она появляется, когда в исходном тексте отсутствует команда для TeX'a «завершить работу» (в LATEX'e эта команда входит в качестве составной части в комплекс действий, выполняемых командой `\end{document}`). Теперь можно вводить с клавиатуры любой текст и команды — TeX отреагирует на них так же, как если бы этот текст и команды присутствовали в вашем файле. Не будем баловаться, а просто наберем `\end{document}` и нажмем на «ввод», после чего трансляция благополучно завершится. Вряд ли вы будете очень часто забывать последнюю строчку в исходном тексте, но иногда, в результате какой-либо сложной ошибки, может случиться так, что TeX «не заметит» команды `\end{document}`, и тогда вы окажетесь лицом к лицу с TeX'овским приглашением-звездочкой.

Наряду с пассивной реакцией на ошибки — все время нажимать на «ввод» или сказать `s` — есть и другая возможность: прямо с клавиатуры вносить исправления в тот текст, который «видит» TeX. На содержимое файла это не повлияет, но изменения в файл можно будет внести и позднее, руководствуясь тем, что записано в log-файле. При этом может сэкономиться время за счет того, что будет меньше «наведенных» ошибок и, как следствие, потребуется меньше прогонов TeX'a для отладки.

Для того чтобы внести исправления с клавиатуры, надо нажать `i` или `I` и затем «ввод». На экране появится такое приглашение:

`insert>`

В ответ на это приглашение следует ввести тот текст и/или команды, которые вы хотите вставить в текст, читаемый TeX'ом. Чтобы продемонстрировать это на практике, давайте приведем файл `test.tex` в исходное состояние, вернув в него последнюю строку `\end{document}`, и еще раз запустим LATEX для его обработки. В ответ на первое же сообщение (по поводу несуществующей команды `\TeXpert`) нажмем `i`, а затем, в ответ на приглашение `insert>`, наберем правильный текст

\TeX pert

и нажмем на «ввод». В ответ на вторую ошибку (когда мы в команде `\end` допустили опечатку в имени окружения `center`) скажем сначала `i`, а затем, в ответ на приглашение, `\end{center}` (кстати, можно делать такие вещи и в один шаг: сразу набрать `i\end{center}` и нажать «ввод»). В ответ на следующую ошибку ничего не остается, как по-прежнему нажать на «ввод»: те знаки в исходном тексте, между которыми должен был стоять пропущенный знак доллара, уже поглощены `\TeX`'ом, и вставить его куда надо в данный момент невозможно; зато в ответ на следующую ошибку (`\lrf` вместо `\left`) наберем `i\left` и нажмем на «ввод». Следующей, «наведенной» ошибки вообще не будет (ведь на сей раз в тексте, который видит `\TeX`, команда `\left` присутствует, а поэтому и на команду `\right` `\TeX` отреагирует правильно); наконец, в ответ на последнюю ошибку опять ничего не остается, кроме как нажать на «ввод»: вставить закрывающую фигурную скобку между `4` и знаком равенства прямо с клавиатуры невозможно. Теперь можно просмотреть, как на сей раз будет выглядеть на печати наш текст; некоторые несуразности наподобие $\sqrt{4} = 2$ в нем останутся, но их будет меньше, чем если бы мы нажимали на «ввод»: не будет потеряно слово «`\TeXpert`», центрированная строка будет действительно центрирована, формула

$$\frac{25}{36} = \left(\frac{1}{1 + \frac{1}{5}} \right)^2$$

будет выглядеть так, как надо. Это существенно, поскольку чем ближе к задуманному получится `dvi`-файл, тем меньше времени мы потратим в дальнейшем на выявление полиграфических недостатков и борьбу с ними (по поводу борьбы с полиграфическими недостатками см. разд. III.7). Теперь остается внести исправления в исходный файл (справляясь с тем, что записано в `log`-файле) и запустить `\LaTeX` вторично, чтобы получить безошибочный текст.

Как мы уже отмечали, в ответ на сообщение об ошибке всегда можно прервать трансляцию, нажав `X` или `x` и «ввод»; кроме того, бывают случаи, когда `\TeX` прерывает трансляцию «по своей инициативе». На практике важны два случая:

- `\TeX` обнаружил 100 ошибок в пределах одного абзаца — тогда выдается сообщение

(*That makes 100 errors; please try again.*)

и трансляция прекращается;

- $\text{\TeX}'$ у не хватило памяти — тогда выдается сообщение наподобие такого:

! \TeX capacity exceeded, sorry [main memory size=65533].

Нехватка памяти может возникнуть в результате таких ошибок, из-за которых \TeX «зацикливается»; тогда достаточно исправить ошибку. Иногда памяти $\text{\TeX}'$ у может действительно не хватить. Так бывает, если в тексте встречаются чудовищно длинные абзацы (длиннее, чем на 4–5 страниц) или сверхсложные таблицы с очень большим количеством строк и столбцов (см. гл. VI по поводу верстки таблиц). Если вы встретились с такой проблемой, то можно проконсультироваться со специалистом (или самому изучить по книге [3]), как использовать \TeX более эффективно (в частности, \TeX можно научить переваривать сколь угодно длинные абзацы). Можно также попробовать найти транслятор $\text{\TeX}'$ а, дающий возможность работать с увеличенным объемом памяти. Возможно, при этом вам понадобится и более мощный компьютер ...

Скажем пару слов про более экзотические способы реакции на ошибки. Во-первых, в ответ на приглашение ? можно набрать h или H и нажать «ввод». В этом случае \TeX выдаст на экран дополнительную информацию по поводу вашей ошибки (вряд ли вы много из нее почерпнете, если вы не \TeX ник), а затем еще раз приглашение ?. Во-вторых, можно набрать r или R (и «ввод», естественно); результат будет такой же, как если бы вы сказали s, с той разницей, что в случае, если аргументом команды \input служит несуществующий файл, никаких вопросов задаваться не будет, а трансляция просто прервется. Наконец, можно набрать Q или q (и «ввод»): результат будет такой же, как от R, с той разницей, что на экран не будет выдаваться вообще ничего (в log-файл все будет записано).

Наконец, режимы реакции на ошибки, задаваемые с клавиатуры с помощью клавиш s, r или q, можно также задать прямо в файле; для этого достаточно в преамбуле дать одну из перечисленных ниже команд.

Команда \scrollmode равносильна нажатию s

Команда \nonstopmode равносильна нажатию r

Команда \batchmode равносильна нажатию q

6. Как читать книгу дальше?

Наш обзор основных понятий завершен, и вы уже можете подготовить с помощью I \TeX 'а несложный текст. Дальнейшее чтение, в зависимости от ваших потребностей, можно построить по-разному: несколько

последующих глав почти независимы друг от друга, а внутри каждой из них материал расположен в порядке возрастания трудности.

В гл. II рассказано про многочисленные тонкости и дополнительные средства, связанные с набором математических формул. Если вас это не очень интересует, можете при первом чтении обращаться к ней только за справками.

Глава III посвящена набору текста «в малом»: в ней рассказывается, в частности, как задавать в тексте шрифты разных начертаний и размеров, как набирать ударения над буквами и специальные типографские значки наподобие знака параграфа, как делать сноски, и т. п.

Глава IV посвящена оформлению текста «в целом»: в ней подробно рассказано про то, какие бывают стили и чем они отличаются друг от друга, как устроить разбиение текста на разделы таким образом, чтобы *LATEX* автоматически создавал заголовки этих разделов и к тому же автоматически их нумеровал, как оформлять титульный лист, как создать оглавление, и тому подобное. Для чтения этой главы подробное знание предыдущей не является обязательным.

В гл. V описаны так называемые псевдорисунки — примитивные картинки, которые можно создавать, не выходя за рамки *LATEX*'а.

Последующие главы посвящены более сложным вопросам. В гл. VI рассказывается о верстке таблиц с помощью *LATEX*'а. В гл. VII объяснено, как можно повысить эффективность своей работы в *LATEX*'е, создавая собственные команды. Первые разделы этой главы можно читать параллельно с гл. II. В гл. VIII рассказывается о таких фундаментальных понятиях *TeX*'а, как «блоки» и «клей»; когда вы усвоите материал этой главы, вы сможете, в частности, создавать некоторые специальные эффекты при верстке более простыми средствами, чем это позволяет собственно *LATEX*. Наконец, заключительная гл. IX, предлагающая знание всего предыдущего материала, рассказывает, как изменить стандартный стиль оформления документов, предоставляемый *LATEX*'ом, применительно к своим нуждам.

Глава II

Как набирать формулы

1. Таблицы спецзнаков с комментариями

В этом разделе мы перечислим все математические знаки, предоставляемые L^AT_EX'ом. Знаков этих очень много, поэтому разобьем их на несколько групп.

1.1. Операции, отношения и просто значки

Начнем с греческих букв. Имя команды, задающей строчную греческую букву, совпадает с английским названием этой буквы (например, буква α задается командой `\alpha`). Исключение составляет буква \circ (она называется «омикрон»): по начертанию она совпадает с курсивной латинской о, так что специальной команды для нее не предусмотрено, и для ее набора достаточно просто написать \circ в формуле. Некоторые греческие буквы имеют по два варианта начертаний; это также отражено в следующей ниже таблице.

α	<code>\alpha</code>	β	<code>\beta</code>	γ	<code>\gamma</code>
δ	<code>\delta</code>	ϵ	<code>\epsilon</code>	ε	<code>\varepsilon</code>
ζ	<code>\zeta</code>	η	<code>\eta</code>	θ	<code>\theta</code>
ϑ	<code>\vartheta</code>	ι	<code>\iota</code>	κ	<code>\kappa</code>
λ	<code>\lambda</code>	μ	<code>\mu</code>	ν	<code>\nu</code>
ξ	<code>\xi</code>	π	<code>\pi</code>	ϖ	<code>\varpi</code>
ρ	<code>\rho</code>	ϱ	<code>\varrho</code>	σ	<code>\sigma</code>
ς	<code>\varsigma</code>	τ	<code>\tau</code>	υ	<code>\upsilon</code>
ϕ	<code>\phi</code>	φ	<code>\varphi</code>	χ	<code>\chi</code>
ψ	<code>\psi</code>	ω	<code>\omega</code>		

Имя команды, задающей прописную греческую букву, пишется с прописной буквы (например, буква Ψ задается командой `\Psi`). Неко-

торые прописные греческие буквы («альфа», например) совпадают по начертанию с латинскими, и для них специальных команд нет — надо просто набрать соответствующую латинскую букву прямым шрифтом (см. с. 54 по поводу того, как это сделать). Не надо использовать греческие буквы Σ и Π из этой таблицы в качестве знаков суммы и произведения: для этих целей есть специальные команды, о которых пойдет речь дальше. Итак, вот прописные греческие буквы, не совпадающие по начертанию с латинскими:

Γ	<code>\Gamma</code>	Δ	<code>\Delta</code>	Θ	<code>\Theta</code>
Λ	<code>\Lambda</code>	Ξ	<code>\Xi</code>	Π	<code>\Pi</code>
Σ	<code>\Sigma</code>	Υ	<code>\Upsilon</code>	Φ	<code>\Phi</code>
Ψ	<code>\Psi</code>	Ω	<code>\Omega</code>		

Как видите, прописные греческие буквы печатаются, в отличие от строчных, прямым шрифтом. Если вам нужны наклонные прописные греческие буквы (вроде Σ), прочтите в разд. 2.3 о том, как их получить в L^AT_EX'е 2.09 (для L^AT_EX'a 2_E — в разд. 2.4).

Следующая серия символов — символы, рассматриваемые T_EX'ом как символы бинарных операций (наподобие знаков сложения, умножения и т. п.). T_EX оставляет в формуле небольшие пробелы по обе стороны этих знаков, кроме случаев, когда есть основания считать, что эти знаки используются не для обозначения операций, а для других целей (если, например, стоят два плюса подряд, то дополнительного пробела между ними не будет). Итак, вот полный список символов бинарных операций:

$+$	$+$	$-$	$-$	$*$	$*$
\pm	<code>\pm</code>	\mp	<code>\mp</code>	\times	<code>\times</code>
\div	<code>\div</code>	\setminus	<code>\setminus</code>	\cdot	<code>\cdot</code>
\circ	<code>\circ</code>	\bullet	<code>\bullet</code>	\cap	<code>\cap</code>
\cup	<code>\cup</code>	\uplus	<code>\uplus</code>	\sqcap	<code>\sqcap</code>
\sqcup	<code>\sqcup</code>	\vee	<code>\vee</code>	\wedge	<code>\wedge</code>
\oplus	<code>\oplus</code>	\ominus	<code>\ominus</code>	\otimes	<code>\otimes</code>
\odot	<code>\odot</code>	\oslash	<code>\oslash</code>	\triangleleft	<code>\triangleleft</code>
\triangleright	<code>\triangleright</code>	\amalg	<code>\amalg</code>	\diamond	<code>\diamond</code>
\wr	<code>\wr</code>	\star	<code>\star</code>	\dagger	<code>\dagger</code>
\ddagger	<code>\ddagger</code>	\bigtriangleup	<code>\bigtriangleup</code>	\bigcirc	<code>\bigcirc</code>
\bigtriangledown	<code>\bigtriangledown</code>				

Обозначения для многих из выписанных знаков длинны и сложны. С этим неудобством борются следующим образом: если в вашем тексте часто встречается какое-то длинное обозначение для математического символа, имеет смысл определить для этого символа свою более

удобную команду (например, `\b{t}{u}` вместо `\bigtriangleup`). Как это сделать, рассказано в начале гл. VII; вы можете прочитать это уже сейчас.

В следующей таблице мы собрали символы «бинарных отношений». Вокруг них TeX также оставляет дополнительные пробелы (не такие, как вокруг символов бинарных операций). Вообще говоря, нет смысла много задумываться об этих пробелах, поскольку TeX оформляет математические формулы в достаточно разумном стиле; о тех случаях, когда размер пробелов в математических формулах приходится корректировать вручную, речь пойдет дальше в этой главе. Команда `\mid` в этой таблице определяет вертикальную черточку, рассматриваемую как знак бинарного отношения; ее *не* следует употреблять, если вертикальная черточка употребляется как аналог скобки (например, как знак абсолютной величины).

<code><</code>	<code><</code>	<code>></code>	<code>></code>	<code>=</code>	<code>=</code>
<code>:</code>	<code>:</code>	<code>\leq</code>	<code>\geq</code>		
<code>\neq</code>	<code>\neq</code>	<code>\sim</code>	<code>\simeq</code>		
<code>\approx</code>	<code>\approx</code>	<code>\cong</code>	<code>\equiv</code>		
<code>\ll</code>	<code>\ll</code>	<code>\gg</code>	<code>\doteq</code>		
<code>\parallel</code>	<code>\parallel</code>	<code>\perp</code>	<code>\in</code>		
<code>\notin</code>	<code>\notin</code>	<code>\ni</code>	<code>\subset</code>		
<code>\subseteq</code>	<code>\subseteq</code>	<code>\supset</code>	<code>\supseteq</code>		
<code>\succ</code>	<code>\succ</code>	<code>\prec</code>	<code>\succcurlyeq</code>		
<code>\preccurlyeq</code>	<code>\preccurlyeq</code>	<code>\asymp</code>	<code>\sqsubseteq</code>		
<code>\sqsubseteq</code>	<code>\sqsubseteq</code>	<code>\models</code>	<code>\vdash</code>		
<code>\dashv</code>	<code>\dashv</code>	<code>\smile</code>	<code>\frown</code>		
<code>\mid</code>	<code>\mid</code>	<code>\bowtie</code>	<code>\propto</code>		
<code>\lhd</code>	<code>\lhd</code>	<code>\unlhd</code>	<code>\rhd</code>		
<code>\unrhd</code>	<code>\unrhd</code>	<code>\sqsubset</code>	<code>\sqsupset</code>		
<code>\Join</code>	<code>\Join</code>				

Последние семь из перечисленных команд (от `\lhd` до `\Join`) в *LaTeX*'е определены только в том случае, если вы подключите стилевой пакет `latexsym`. См. с. 23 по поводу того, как подключать стилевые пакеты.

В следующей таблице собраны стрелки различных видов. Среди них есть и вертикальные; как их использовать при наборе формул, вы узнаете попозже, в разд. 3.2.

<code>\rightarrow</code>	<code>\rightarrow</code>	<code>\longrightarrow</code>	<code>\Rightarrow</code>	<code>\Rrightarrow</code>
<code>\Longrightarrow</code>	<code>\Longrightarrow</code>	<code>\hookrightarrow</code>	<code>\hookrightarrow</code>	

→	\mapsto	→	\longmapsto	~	\leadsto
←	\gets	←	\longleftarrow	⇐	\Leftarrow
⟵	\Longleftarrow	⟵	\hookleftarrow		
↔	\leftrightarrow	↔	\longleftrightarrow		
↔	\Leftrightarrow	↔	\Longleftrightarrow		
↑	\uparrow	↑	\Uparrow		
↓	\downarrow	↓	\Downarrow		
↕	\updownarrow	↕	\Updownarrow		
↗	\nearrow	↘	\searrow		
↖	\swarrow	↖	\nwarrow	←	\leftharpoondown
↖	\leftharpoonup	→	\rightharpoonup		
→	\rightharpoondown	=	\rightleftharpoons		

ε Команда \leadsto в \LaTeX^e будет определена только если подключить стилевой пакет latexsym.

ε Из привычных российскому читателю символов в вышеприведенных таблицах нет знаков \geq и \leq , более привычных, чем \geq и \leq ; кроме того, греческая буква «каппа» лучше смотрится в виде κ , чем в виде κ (\kappa). Эти символы отсутствуют в «классическом» $\text{\LaTeX}^{\text{овском}}$ наборе; при использовании $\text{\LaTeX}^{\text{ом}}$ они становятся доступными, если подключить стилевой пакет amssymb. Как объяснялось на с. 23, для этого надо после строчки с командой \documentclass написать

\usepackage{amssymb}

(если в вашем файле уже есть команда \usepackage, то можно добавить слово amssymb через запятую в ее аргумент). При условии, что это сделано, можно задавать в математических формулах букву κ командой \varkappa, а символы \leq и \geq — командами \leqslant и \geqslant соответственно.

1.2. Операции с пределами и без

В следующей таблице собраны «функции типа синус» — команды для воспроизведения названий математических операций наподобие sin, log и т. п., обозначаемых последовательностью букв, набираемых прямым шрифтом. Любую из этих операций можно снабдить верхним и/или нижним индексом (см. пример на с. 32).

log	\log	lg	\lg	ln	\ln
arg	\arg	ker	\ker	dim	\dim

<code>hom</code>	<code>\hom</code>	<code>deg</code>	<code>\deg</code>	<code>exp</code>	<code>\exp</code>
<code>sin</code>	<code>\sin</code>	<code>arcsin</code>	<code>\arcsin</code>	<code>cos</code>	<code>\cos</code>
<code>arccos</code>	<code>\arccos</code>	<code>tan</code>	<code>\tan</code>	<code>arctan</code>	<code>\arctan</code>
<code>cot</code>	<code>\cot</code>	<code>sec</code>	<code>\sec</code>	<code>csc</code>	<code>\csc</code>
<code>sinh</code>	<code>\sinh</code>	<code>cosh</code>	<code>\cosh</code>	<code>tanh</code>	<code>\tanh</code>
<code>coth</code>	<code>\coth</code>				

В этой таблице обозначения `tan`, `arctan` и т. д. — не что иное, как принятые в англоязычной литературе обозначения для тангенса, арктангенса и т. д. В отечественной литературе, однако же, принято обозначать `tg`, `ctg` и т. д. Так как в стандартном комплекте TeX'a или LATEX'a команд для этого нет, их приходится, при необходимости, определять самому. Это просто: в преамбуле документа надо написать

```
\newcommand{\tg}{\mathop{\rm tg}\nolimits}
```

После этого команда `\tg` будет создавать в математической формуле запись `tg` с правильными пробелами вокруг нее. Другие команды такого типа определяются аналогично, надо только вместо `tg` написать то название функции (скажем, `arctg`), которое должно появиться на печати. Если вы получили LATEX' вместе с русификацией, то не исключено, что в ней уже определены команды для принятых в России обозначений тангенса, арктангенса и т. п. (по крайней мере они есть в той русификации, которая использовалась при верстке этой книги).

Описанный выше способ определения команд для «функций типа синус» является частным случаем существующей в LATEX'е конструкции для определения новых команд (см. гл. VII).

Еще один символ, который принято набирать прямым шрифтом, — это символ `mod`, который используется в записи «сравнений по модулю». Обычно он употребляется не сам по себе, а в сочетании со знаком `≡` (см. пример ниже); в этом случае для записи сравнения удобна команда `\pmod`, которой пользуются так:

Легко видеть, что $23^{1993} \equiv 1 \pmod{11}$. Легко видеть, что

$$\$23^{1993} \backslash equiv 1 \pmod{11}\$.$$

Обратите внимание, что скобки вокруг `mod 11` получаются автоматически; правая часть сравнения — весь текст, заключенный между `\equiv` и `\pmod`. Иногда, кроме того, символ `mod` используется как символ бинарной операции, например, так:

$f_*(x) = f(x) \bmod G$

$\$f_*(x)=f(x)\bmod G\$$

Как видно из примера, в этом случае надо писать `\bmod`.

Теперь обсудим, как можно было бы получить, скажем, формулу

$$\sum_{i=1}^n n^2 = \frac{n(n+1)(2n+1)}{6}$$

с дополнительными элементами над и под знаком операции. В данной формуле эти элементы называются «пределы суммирования», поэтому в ТЕХнической терминологии записи над и под знаком операции принято называть «пределами» (по-английски `limits`). В исходном тексте «пределы» обозначаются точно так же, как индексы; имея в виду, что знак суммы генерируется командой `\sum`, получаем, что вышеназванную формулу можно получить так:

```
$$
\sum_{i=1}^n n^2=\frac{n(n+1)(2n+1)}{6}
$$
```

В этом примере существенно, что формула была выключной; во внутритекстовой формуле «пределы» печатаются на тех же местах, что и индексы:

Тот факт, что $\sum_{i=1}^n (2n - 1) = n^2$,
следует из формулы для суммы
арифметической прогрессии.

Тот факт, что
 $\sum_{i=1}^n (2n-1)=n^2$,
следует из формулы для суммы
арифметической прогрессии.

(можно добиться, чтобы они и во внутритекстовой формуле были сверху и снизу — см. ниже). Вот список операций, ведущих себя так же, как `\sum`:

\sum	<code>\sum</code>	\prod	<code>\prod</code>	\bigcup	<code>\bigcup</code>
\bigcap	<code>\bigcap</code>	\coprod	<code>\coprod</code>	\bigoplus	<code>\bigoplus</code>
\otimes	<code>\bigotimes</code>	\bigodot	<code>\bigodot</code>	\bigvee	<code>\bigvee</code>
\wedge	<code>\bigwedge</code>	\biguplus	<code>\biguplus</code>	\bigsqcup	<code>\bigsqcup</code>
\lim	<code>\lim</code>	\limsup	<code>\limsup</code>	\liminf	<code>\liminf</code>
\max	<code>\max</code>	\min	<code>\min</code>	\sup	<code>\sup</code>
\inf	<code>\inf</code>	\det	<code>\det</code>	\Pr	<code>\Pr</code>
\gcd	<code>\gcd</code>				

Все обозначения из этой таблицы употребительны и в отечественной литературе, за исключением `\gcd` для наибольшего общего делителя (у нас его иногда обозначают н.о.д.) и `\Pr` для вероятности, обычно обозначаемой P .

Еще одна «математическая операция», для которой требуются «пределы», — это интеграл. В \LaTeX 'е есть команды \int для обычного знака интеграла \int и \oint для знака «контурного интеграла» \oint . При этом, для экономии места, пределы интегрирования помещаются не сверху и снизу от знаков интеграла, а по бокам (даже и в выключных формулах):

$$\int_0^1 x^2 dx = 1/3 \quad \begin{array}{c} \$\$ \\ \text{\int_0^1x^2\,dx=1/3} \\ \$\$ \end{array}$$

Если, тем не менее, необходимо, чтобы пределы интегрирования стояли над и под знаком интеграла, то надо непосредственно после \int записать команду \limits , а уже после нее — обозначения для пределов интегрирования:

$$\int_0^1 x^2 dx = 1/3 \quad \begin{array}{c} \$\$ \\ \text{\int\limits_0^1 x^2 dx=1/3} \\ \$\$ \end{array}$$

Тот же прием с командой \limits можно применить, если хочется, чтобы во внутритекстовой формуле «пределы» у оператора стояли над и под ним, а не сбоку.

Если, с другой стороны, надо, чтобы «пределы» у какого-либо оператора стояли не над и под знаком оператора, а сбоку, то после команды для знака оператора надо записать команду \nolimits , а уже после нее — обозначения для «пределов»:

$$\prod_{i=1}^n i = n! \quad \begin{array}{c} \$\$ \\ \text{\prod\nolimits_{i=1}^n i=n!} \\ \$\$ \end{array}$$

1.3. Разное

Мы уже перечислили почти все символы, используемые \LaTeX 'ом в математических формулах. Остались скобки различных видов (им будет посвящен специальный разд. 2.6), а также ряд значков (среди них есть и часто встречающиеся), не входящих ни в какой из разделов нашей классификации. Они собраны в следующей таблице.

∂	\partial	\triangle	\triangle	\angle	\angle
∞	\infty	\forall	\forall	\exists	\exists
\emptyset	\emptyset	\neg	\neg	\aleph	\aleph
$/$	\prime	\hbar	\hbar	∇	\nabla

<i>i</i>	\imath	<i>j</i>	\jmath	<i>l</i>	\ell
✓	\surd	♭	\flat	#	\sharp
▮	\natural	▮	\top	⊥	\bot
wp	▮	▮	\Re	▮	\Im
＼	\backslash		\mid	♠	\spadesuit
♣	\clubsuit	◊	\diamondsuit	♥	\heartsuit
Ⓜ	\mho	□	\Box	◇	\Diamond
†	\dag	§	\S	©	\copyright
‡	\ddag	¶	\P	£	\pounds

Последние шесть символов (от † до £) можно использовать не только в формулах, но и в тексте.

Символ \emptyset (\emptyset) — это, конечно, обозначение для пустого множества. В отечественной литературе более принято другое начертание для этого символа: \varnothing . При использовании L^AT_EX'ом 2_ε символ пустого множества в этом начертании задается командой \varnothing; чтобы можно было пользоваться этой командой, необходимо подключить стилевой пакет amssymb.

Не следует смешивать команды \parallel и \| . На печати они дают один и тот же значок ||, но с разными пробелами (полиграфист бы сказал: «отбивками») вокруг него. Команда \parallel нужна для обозначения бинарного отношения «параллельность», в то время как \| — это один из видов скобок:

В школьных учебниках геометрии встречаются такие формулы, как $AB \parallel CD$.

В университетских учебниках анализа часто пишут, что $\|A\| = \sup(|Ax|/|x|)$.

В школьных учебниках геометрии встречаются такие формулы, как \$AB\parallel CD\$.

В университетских учебниках анализа часто пишут, что \$\|A\|=\sup(|Ax|/|x|)\$.

ε Команды \mho, \Box и \Diamond будут доступны в L^AT_EX'е 2_ε только если подключить стилевой пакет latexsym.

Символы, обозначаемые командами \imath и \jmath, нужны для постановки дополнительных значков над буквами *i* и *j* (об этом пойдет речь в разд. 2.8).

Команды \nabla и \bigtriangledown задают совершенно разные символы, и их не надо путать. Обратите также внимание на символ, задаваемый командой \prime. Это — тот самый штрих, который получается, если после символа в формуле поставить знак ' ; на самом деле записи x' и x^{\prime} \prime совершенно равносильны.

Наша последняя таблица — таблица синонимов. В ней представлены математические символы, которые можно набирать двумя различными способами:

*	* или \ast
\neq	\neq или \neq
\leq	\leq или \leq
\geq	\geq или \geq
[[или \lbrack
]] или \rbrack
{	\{ или \lbrace
}	\} или \rbrace
\rightarrow	\rightarrow или \rightarrow
\leftarrow	\leftarrow или \leftarrow
\ni	\ni или \owns
\wedge	\wedge или \wedge
\vee	\vee или \vee
\neg	\neg или \not
\parallel	\parallel или \parallel

1.4. Какие еще есть символы

В наших таблицах собрано более полутора сотен математических символов, не считая операций типа \log или \sin . Тем не менее, для набора формул этого мало. Во-первых, в формулах встречаются скобки разнообразных начертаний (и размеров). О них речь пойдет в разд. 2.6. Во-вторых, часто бывает нужно, чтобы обозначения переменных в формуле печатались не курсивом, как это делается в \TeX 'е по умолчанию, а другим шрифтом. Как этого добиться, рассказано в разд. 2.3. И, наконец, в тексте, который вы набираете, может встретиться значок, которого нет в таблицах этой главы. Если вы работаете в \LaTeX 'е 2 ϵ , то попробуйте поискать его в таблицах приложения Д. Если он там присутствует, то вы сможете им воспользоваться. При этом надо подключить стилевой пакет *amssymb*, о котором неоднократно упоминалось выше.

2. Важные мелочи

2.1. Нумерация формул

В математических текстах обычно приходится для удобства ссылок нумеровать формулы. \TeX позволяет организовать эту нумерацию таким образом, чтобы номера формул и ссылки на них генерировались ав-

томатически (см. разд. I.2.11). Нумеровать таким образом можно только **выключные** формулы. Делается это так.

Выключная формула, которую вы нумеруете, должна быть оформлена как окружение `equation` (знаков `$$` быть не должно!). Каждая такая формула на печати автоматически получит номер. Чтобы на него можно было ссылаться, надо формулу пометить: в любом месте между `\begin{equation}` и `\end{equation}` поставить команду `\label`, и после этого команда `\ref` будет генерировать номер формулы (см. с. 26). Поясним все сказанное примером:

Каждый школьник должен знать, что

$$7 \times 9 = 63. \quad (1)$$

.....

Из формулы (1) следует, что
 $63/9 = 7$.

Каждый школьник должен знать, что

```
\begin{equation}
\label{trivial}
7\times9=63.
\end{equation}
.....
```

Из формулы `\ref{trivial}` следует, что $63/9=7$$.

Знак `"` мы поставили, чтобы номер формулы и слово «формулы» не попали на разные строки (см. с. 89). Обратите внимание, что скобки вокруг номера формулы, сгенерированного командой `\ref`, автоматически не ставятся.

Можно также использовать команду `\pageref` вместо `\ref` — тогда на печати получится не номер формулы, а номер страницы, на которую попала эта формула.

То, как именно выглядит на печати номер формулы, зависит от стиля документа (см. с. 19, 141): например, в стиле `article` (статья) формулы имеют сплошную нумерацию, а в стиле `book` (книга) нумерация формул начинается заново в каждой главе, и номер, скажем, формулы 5 из главы 3, генерируемый окружением `equation`, имеет вид (3.5). В разделе, посвященном модификации стандартных стилей, мы расскажем, как можно самостоятельно менять вид номеров формул.

Кроме того, вы можете вообще не пользоваться автоматической генерацией номеров формул, а ставить их вручную. Чтобы номер выглядел при этом красиво, удобно воспользоваться ТЕХ'овской командой `\eqno`. Следующий пример показывает, как это делать:

Простое тождество

$$7 \times 9 = 63 \quad (3.2)$$

известно каждому школьнику.

Простое тождество

`$$`

$$7\times9=63\eqno (3.2)$$

`$$`

известно каждому школьнику.

Выключная формула, нумеруемая с помощью команды `\eqno`, должна быть оформлена с помощью знаков `$$`; номером формулы будет служить весь текст, заключенный между `\eqno` и закрывающими формулу `$$`; этот текст обрабатывается \TeX 'ом так же, как математические формулы (стало быть, пробелы игнорируются, буквы печатаются «математическим курсивом», и т. п.). Можно также вместо `\eqno` сказать `\leqno`, тогда ваш номер формулы будет не справа, а слева:

Менее простое тождество

$$(*) \quad \sin^2 x + \cos^2 x = 1$$

известно каждому десяти-
класснику.

Менее простое тождество

`$$`

$$\backslash\sin^2x+\backslash\cos^2x=1$$

`\leqno (*)`

`$$`

известно каждому десятикласснику.

Никаких автоматических ссылок на формулу, генерируемую командой `\eqno` или `\leqno`, \TeX не создает, и в этом случае за корректность ссылок отвечаете только вы.

2.2. Переносы в формулах

При необходимости \TeX может перенести часть внутритекстовой формулы на другую строчку. Такие переносы возможны после знаков «бинарных отношений», наподобие знака равенства⁴ (см. с. 45) или «бинарных операций», наподобие знаков сложения или умножения (см. с. 44), причем последний знак в строке, вопреки российской традиции, не дублируется в начале следующей. Чтобы избежать таких переносов, можно воспользоваться тем обстоятельством, что \TeX не разрывает при переносе часть формулы, заключенную в фигурные скобки. В частности, можно заключить в фигурные скобки всю формулу, в которой произошел нежелательный перенос, от открывающего ее знака доллара до закрывающего: после этого можете быть уверены, что переноса этой формулы ни при каких обстоятельствах не произойдет.

Вышеописанный способ борьбы с неудачными переносами в формулах имеет один недостаток: при этом затрудняется верстка абзацев и возрастает вероятность появления неприятных сообщений `Overfull \hbox` (см. разд. III.7).

Более гибкий способ борьбы с переносами в формулах — записать в преамбуле файла строку

`\binoppenalty=10000`

⁴Стрелки также рассматриваются \TeX 'ом как бинарные отношения.

и/или строку

`\relpenalty=10000`

Первая из этих строк запретит все разрывы строк после знаков бинарных операций, а вторая — после знаков бинарных отношений, и при этом помех верстке абзаца будет меньше, чем при заключении всей формулы в фигурные скобки.

Для любознательных поясним, что `\binoppenalty` и `\relpenalty` — параметры (TeX'овские), значением которых может быть целое число (см. с. 22 по поводу TeX'овских параметров). Эти параметры определяют степень нежелательности разрыва строки после символов бинарной операции и бинарного отношения соответственно (чем больше значение соответствующего параметра, тем менее желателен разрыв строки). По умолчанию значение `\binoppenalty` равно 700, а значение `\relpenalty` равно 500. Можно присвоить им в преамбуле большие значения, тогда вероятность разрывов уменьшится. 10000 означает абсолютный запрет.

При заключении всей формулы в фигурные скобки верстка абзацев затрудняется потому, что TeX лишается возможности варьировать в ней интервалы между символами для выравнивания строк (см. разд. III.7).

Выключные формулы, в отличие от внутритекстовых, TeX никогда не переносит. Если выключная формула не помещается в строку, то при трансляции вы получите сообщение «*Overfull \hbox*» (в разд. III.7 подробно рассказано, в каких еще ситуациях выдается такое сообщение), и вам придется разбить формулу на строки вручную. Как это делать, мы объясним в разд. 3.4.

2.3. Смена шрифтов в формуле: *ЛaTeX 2.09*

Как уже говорилось, по умолчанию все латинские буквы в формулах набираются курсивным шрифтом. Если требуется иной шрифт, можно дать команду переключения на него. В разд. I.2.5 приведено несколько команд, изменяющих начертание шрифта; их полный список содержится в разд. III.4 (используйте в формулах только команды, меняющие начертание; для ручного управления размерами символов в формулах предназначены команды, описанные в разд. 4.2). В формулах можно использовать команды `\rm`, `\bf`, `\sf`, `\tt` и `\sl`.

ε В *ЛaTeX*'е 2ε команда `\sl` в формулах запрещена.

Чаще всего используются такие команды, как `\rm` для переключения на прямой шрифт и `\bf` для переключения на полужирный шрифт. Если команда переключения шрифта дана внутри группы, то после выхода из группы шрифт, как обычно, восстанавливается:

Обозначение P^n используется для проективного пространства.

Обозначение $\{\bf P\}^n$ используется для проективного пространства.

Команда `\mit` возвращает шрифт к «математическому курсиву», принятому по умолчанию; нужда в ней возникает редко. Один случай, когда эта команда реально полезна, — это если мы хотим получить прописные греческие буквы, напечатанные наклонным шрифтом. В этом случае надо дать (внутри группы) команду `\mit` и команду для соответствующей греческой буквы:

$$\Sigma_a^X = C$$

$$\{\mit\Sigma\}^X_a=C$$

Наряду со шрифтами, применяемыми в текстах, L^AT_EX предоставляет еще «рукописный» шрифт, который можно употреблять только в формулах. Этим шрифтом можно набирать только прописные латинские буквы; переключение на него задается командой `\cal`. Вот пример:

Касательное расслоение к многообразию X обозначается или так: T_X , или так: T_X .

Касательное расслоение к многообразию X обозначается или так: $\{\cal T\}_X$, или так: T_X .

Символ `` `` в этих примерах — это «символ неразрывного пробела» (см. с. 89); мы поставили его, чтобы строка не начиналась с формулы.

Все перечисленные команды смены шрифта в формулах действуют только на латинские буквы и не оказывают влияния на начертание греческих букв (кроме прописных) и спецзнаков.

Теперь, когда вы знаете, как печатать символы в формулах прямым шрифтом, может возникнуть искушение восполнить отсутствие в стандартном комплекте L^AT_EX'a команды, дающей функцию `tg`, путем набора чего-нибудь вроде `\rm tg` x . Так делать, однако, не надо, поскольку при этом пробелы будут неправильными:

В формуле $\rm tg x$ буква x слишком близка к знаку тангенса. А вот в формуле $\sin x$ пробелы правильные.

В формуле $\{\rm tg\} x$ буква x слишком близка к знаку тангенса. А вот в формуле $\{\sin x$ пробелы правильные.

Правильно действовать так, как рекомендуется на с. 47. Если вам хочется узнать, почему все так получается, прочтите разд. 4.4.

Можно сделать так, чтобы по умолчанию латинские буквы в формулах набирались не курсивом, а полужирным прямым шрифтом. Для этого надо в преамбуле документа поставить команду `\boldmath`.

Таблица II.1. Смена шрифтов в формуле в $\text{\LaTeX}'e 2_\epsilon$

В $\text{\LaTeX}'e 2.09:$	В $\text{\LaTeX}'e 2_\epsilon$ рекомендуется:	Получается:
<code>{\rm x}+y</code>	<code>\mathrm{x+y}</code>	$x + y$
<code>{\bf x}+y</code>	<code>\mathbf{x+y}</code>	$\mathbf{x} + y$
<code>{\sf x}+y</code>	<code>\mathsf{x+y}</code>	$\mathsf{x} + y$
<code>{\tt x}+y</code>	<code>\mathtt{x+y}</code>	$\mathtt{x} + y$
<code>{\cal T}_X</code>	<code>\mathcal{T}_X</code>	\mathcal{T}_X
<code>{\mit \Gamma}+\Delta</code>	<code>\mathit{\Gamma\Delta}+\Delta</code>	$\mathit{\Gamma} + \Delta$

Если вы хотите включить в формулу какой-либо текст, то одной команды `\rm` для этого отнюдь не достаточно: любой текст, пусть даже он набирается прямым шрифтом, \TeX рассматривает как часть математической формулы, и в соответствии с этим игнорирует те пробелы, которые ставите вы, и расставляет пробелы по собственным правилам:

$$\sqrt{x^3} = x \text{ для всех } x. \quad \begin{array}{l} \$\$ \\ \text{\sqrt\{x^3\}}=x \text{ \{\\rm для всех\} } x. \\ \$\$ \end{array}$$

Что на самом деле надо сделать, чтобы вставить текст в формулу, описано в разд. 2.5.

- ε В $\text{\LaTeX}'e 2_\epsilon$ существуют и другие команды для смены шрифтов в формулах, а также возможность использования в формулах дополнительных шрифтов. Обо всем этом — в следующем разделе.

2.4. Смена шрифтов в формуле: $\text{\LaTeX}'e 2_\epsilon$

Смена шрифтов в формуле организована в $\text{\LaTeX}'e 2_\epsilon$ иначе, чем в $\text{\LaTeX}'e 2.09$. Все перечисленные в предыдущем разделе команды, кроме `\sl`, в $\text{\LaTeX}'e 2_\epsilon$ поддерживаются (пока, по крайней мере) из соображений совместимости, но наряду с ними определены и рекомендуются к использованию команды смены шрифта, действующие на иных принципах: вместо переключения на новый шрифт во всей формуле, действующего до нового переключения шрифта (или до закрывающей фигурной скобки, если команда переключения шрифта была дана из группы) применяются команды, действующие только на одну непосредственно следующую букву. Эти команды собраны в табл. II.1; для сравнения мы указываем, как то же самое делается в $\text{\LaTeX}'e 2.09$. Подчеркнем еще раз, что изменились не только названия (`\mathrm` вместо `\rm` и т. д.), но и принцип действия команд.

В последней строке таблицы показано, как можно получить прописные греческие буквы в наклонном начертании с помощью команды `\mathit`. При использовании стилевым пакетом `amsmath`, открывающим доступ к возможностям *AMS-TeX*'а, этот способ не работает, и наклонные прописные греческие буквы следует задавать по-иному. См. с. 337.

Если нужно, чтобы другим шрифтом была напечатана не одна буква, а несколько, надо все эти буквы взять в фигурные скобки:

Множество особенностей многообразия X обозначается X_{sing} .

Множество особенностей многообразия X обозначается $X_{\{\mathit{sing}\}}$.

Все сказанное означает, что команда `\mathit` и ей подобные принимают один обязательный аргумент — фрагмент формулы, который надо напечатать другим шрифтом. На первый взгляд, это противоречит сказанному на с. 23: ведь обязательный аргумент должен быть в фигурных скобках, а в конструкциях вроде `\mathbf{x}` никаких фигурных скобок нет. Дело в том, что, в дополнение к сказанному на с. 23, действует еще одно правило: если после имени команды, принимающей обязательный аргумент, следует не открывающая фигурная скобка, а буква, то в качестве аргумента будет воспринята именно эта буква. Так что можно было бы писать `\mathbf{x}` вместо `\mathbf{x}`, но так обычно не делают, чтобы не нажимать лишний раз на клавиши. Ср. с. 93.

Если подключить стилевой пакет `amsfonts` (последний раз напомним, что для этого достаточно после команды `\documentclass` вставить в файл строку

```
\usepackage{amsfonts}
```

или же, если команды `\usepackage` в преамбуле уже присутствуют, внести слово `amsfonts` через запятую в аргумент любой из них), то в математических формулах можно использовать еще два шрифта: ажурный ($\mathbb{R}, \mathbb{C}, \mathbb{Q} \dots$) и готический ($\mathfrak{S}, \mathfrak{p}, \mathfrak{g} \dots$). Ажурным шрифтом можно печатать только прописные буквы; он задается командой `\mathbb` (как и в случае с остальными командами, описываемыми в этом разделе, ажурным шрифтом печатается буква, следующая непосредственно после команды `\mathbb`; если надо напечатать этим шрифтом несколько букв, их следует взять в фигурные скобки). Готический шрифт задается командой `\mathfrak`; она также действует только на непосредственно следующую букву (или на несколько букв, если они взяты в фигурные скобки):

Алгебра $\mathfrak{sl}_2(\mathbb{C})$ играет особую роль в теории представлений.

Алгебра $\mathfrak{sl}_2(\mathbb{C})$ играет особую роль в теории представлений.

Существует также стилевой пакет `eufrak`, при подключении которого становится доступным готический шрифт (с командой `\mathfrak{}`), но не ажурный.

Наконец, есть возможность использовать в формулах вариант рукописного шрифта, в котором буквы имеют более изысканные очертания:

А В С Д Е Ф...

Для этого надо подключить стилевой пакет `euscript`; команда, задающая этот шрифт, называется `\EuScript` (не исключено, что к моменту выхода книги в свет ее переименуют).

Остается только напомнить две вещи. Во-первых, ажурный и готический шрифты, о которых шла речь сейчас, можно использовать только в формулах, и набирать с их помощью обычный текст невозможно (так же, как невозможно набирать греческий текст с помощью команд `\alpha`, `\beta` и т. д.). И во-вторых, все возможности, описанные в этом разделе, доступны в $\text{\LaTeX}'e 2\varepsilon$, а не в $\text{\LaTeX}'e 2.09$.

2.5. Включение текста в формулы

В математическую формулу можно включить фрагмент обычного текста с помощью $\text{\TeX}'$ овской команды `\mbox`. В следующем примере продемонстрировано, как это можно сделать; в нем используется еще команда `\quad`, делающая в тексте или формуле пробел размером 2em (см. с. 25 по поводу единиц длины, применяемых $\text{\TeX}'$ ом); подробнее по поводу команд, создающих пробелы в формулах, см. разд. 4.1; по поводу команд, создающих пробелы в тексте, см. разд. III.2.3.

$$\sqrt{x^3} = x \quad \text{для всех } x. \quad \begin{aligned} &\$ \$ \\ &\backslash\sqrt{x^3}=x\backslash\quad \\ &\backslash\mbox{для всех } x. \\ &\$ \$ \end{aligned}$$

Аргумент команды `\mbox` обрабатывается $\text{\TeX}'$ ом как обычный текст: пробелы не игнорируются, слова набираются не математическим курсивом, а тем же шрифтом, который был текущим перед началом формулы (у нас это был обычный прямой шрифт; если вы хотите, чтобы шрифт был другой, можно внутри аргумента команды `\mbox` дать команду смены шрифта). Весь текст, являющийся аргументом команды `\mbox`, будет напечатан в одну строку. В приведенном примере мы оставили пробел перед закрывающей фигурной скобкой, чтобы обеспечить пробел между текстом и формулой (фрагмент текста, созданный командой `\mbox`, рассматривается $\text{\TeX}'$ ом как одна большая буква; пробел в формуле между «буквой», содержащей текст из `\mbox`, и буквой x

будет недостаточен). Команда `\qquad` была использована по аналогичной причине.

На самом деле можно было бы написать даже так:

```
$$
\sqrt{x^3}=x\qquad\text{мбок для всех $x$.}
$$
```

Аргумент команды `\mbox` рассматривается как текст, но этот текст вполне может, в свою очередь, содержать формулы!

При включении текста в формулы с помощью команды `\mbox` важно иметь в виду вот что. Как читатель, видимо, уже заметил, в математических формулах верхние и нижние индексы, числитель и знаменатель дробей, созданных с помощью команды `\frac`, и тому подобные фрагменты набираются более мелким шрифтом, чем остальная часть формулы. Однако в тексте, созданном с помощью `\mbox`, размер шрифта не изменится, в какую бы часть формулы этот текст не попал.

Команда `\mbox` еще встретится нам в следующей главе, когда речь пойдет о предотвращении переносов в словах; более подробно мы ее рассмотрим в главе о «блоках».

2.6. Скобки переменного размера

Если заключенный в скобки фрагмент формулы занимает много места по вертикали (за счет дробей, степеней и тому подобного), то и сами скобки должны быть большего размера, чем обычные. В TeX'е на этот случай предусмотрен механизм автоматического выбора размера скобок. Пользуются им так.

В формуле

$$e = \lim_{n \rightarrow \infty} \left(1 + \frac{1}{n} \right)^n .$$

скобки обычного размера вокруг $1 + \frac{1}{n}$ смотрелись бы плохо; поэтому при ее наборе надо поставить команду `\left` перед открывающей скобкой и команду `\right` перед закрывающей:

```
$$
e=\lim_{n \rightarrow \infty}
\left( 1 + \frac{1}{n} \right)^n
$$
```

Если перед одной скобкой стоит `\left`, а перед другой скобкой стоит `\right`, то на печати размер этих скобок будет соответствовать высоте фрагмента формулы, заключенного между `\left` и `\right`.

Конструкция с `\left` и `\right` применима не только к круглым скобкам. В следующей таблице перечислены скобки и некоторые другие символы, которые с помощью `\left` и `\right` автоматически принимают нужный размер. ТЕХНИЧЕСКИЙ термин для таких символов — ограничители (по-английски *delimiters*).

(())	[[-
]]	{	\{	}	\}	
\lfloor	\rfloor	\lfloor	\rfloor	\lfloor	\lfloor	\lceil
\rceil	\langle	\langle	\rangle	\rangle	\rangle	\rangle
		\	\	/	/	/
\backslash						

Вместо `\left\langle` можно писать `\left<`, и аналогичным образом вместо `\right\rangle` можно писать `\right>` (однако же `<` — это не `\langle`!). Кроме знаков, перечисленных в этой таблице, менять свои размеры под действием `\left` и `\right` могут и вертикальные стрелки из таблицы на с. 45.

Вместе с каждой командой `\left` в формуле должна присутствовать соответствующая ей команда `\right`, в противном случае ТЕХ выдаст сообщение об ошибке. Вместе с тем ТЕХ вовсе не требует, чтобы «ограничители» (например, скобки) при командах `\left` и `\right` были расположены сколько-нибудь осмысленно с математической точки зрения: вы вполне можете написать что-нибудь вроде `\left(...\right]`, или даже, вопреки смыслу слов *left* и *right*, `\left)... \right(` — за правильность своих формул отвечаете только вы, и ТЕХ тут вам не помощник.

Вместо «ограничителя» после команды `\left` или `\right` можно поставить точку. На месте этой точки ничего не напечатается, а другой «ограничитель» будет необходимого размера. Вот два примера того, как можно использовать этот прием. Во-первых, таким способом можно создать косую дробную черту увеличенного размера (символ `/` также является «ограничителем» — см. таблицу):

$$M(f) = \left(\int_a^b f(x) dx \right) / (b - a)$$

\$\$
 M(f) = \left. \left(\int_a^b f(x) dx \right) \right/ (b - a)
 \left. \left(\int_a^b f(x) dx \right) \right/ (b - a)

В этом примере используется пока неизвестная вам команда `\,`, создающая дополнительный маленький пробел между $f(x)$ и dx — это один из немногих случаев, когда TeX не может автоматически создать требуемые пробелы, и ему надо помочь. Подробнее о таких вещах речь пойдет ниже, в разд. 4.1. Другой пример использования ограничителя без пары таков:

$$\int_a^b \frac{1}{2}(1+x)^{-3/2} dx = -\frac{1}{\sqrt{1+x}} \Big|_a^b$$

```
$$
\begin{aligned}
&\backslash int \backslash limits_a^b \backslash frac{1}{2} \\
&(1+x)^{-3/2} dx = \\
&\backslash left .-\backslash frac{1}{\backslash sqrt{1+x}} \\
&\backslash right |_a^b
\end{aligned}
$$
```

Здесь, кстати, мы не поставили `\,` перед `dx`, поскольку необходимое свободное место возникает за счет показателя степени.

Наконец, важный пример использования ограничителей без пары — использование их для набора систем уравнений, о чем пойдет речь в разд. 3.2.

До сих пор у нас речь шла только о том, что размеры ограничителей выбираются автоматически с помощью команд `\left` и `\right`; бывают, однако, ситуации, когда такой автоматический выбор размера приводит к неудовлетворительным результатам или даже вообще невозможен. Вот, например, ситуация, когда `\left` и `\right` не срабатывают:

$$||x+1| - |x-1|| \quad \$\backslash left| \backslash x+1| - \backslash x-1| \backslash right| \$$$

Для удобочитаемости этого выражения хотелось бы, чтобы внешние знаки модуля были выше, чем внутренние, но этого не получается: поскольку в формуле выступающих элементов нет, то и команды `\left` и `\right` не считают нужным увеличить ограничители, в которые формула заключена.

А иногда бывает так, что автоматически получающиеся ограничители слишком велики. В следующем примере совсем не обязательно, чтобы скобки охватывали и пределы суммирования, что получается при использовании `\left` и `\right`:

$$\left(\sum_{k=1}^n x^k \right)^2 \quad $$
\begin{aligned}
&\backslash left(\\
&\backslash sum_{k=1}^n x^k \\
&\backslash right)^2
\end{aligned}
$$$$

Во всех этих случаях имеет смысл указать размер ограничителя явно. Для этого предусмотрены \TeX'овские команды $\backslash\bigl$, $\backslash\Bigl$, $\backslash\biggl$ и $\backslash\Biggl$ для левых ограничителей и $\backslash\biggr$, $\backslash\Bigr$, $\backslash\bigggr$ и $\backslash\Bigggr$ для правых ограничителей. Мы перечислили эти команды в порядке возрастания размера создаваемого ими ограничителя. В частности, приведенные выше примеры выглядели бы лучше, если бы мы в примере со знаками модуля написали так:

$$\left| |x+1| - |x-1| \right| \quad \$\backslash\Bigl| \ |x+1|-|x-1|\backslash\Bigr| \$$$

а пример со знаком суммы кому-то мог бы понравиться больше, если бы мы написали так:

$$\left(\sum_{k=1}^n x^k \right)^2 \quad \$\$ \\ \backslash\Bigl(\\ \backslash\sum_{k=1}^n x^k \\ \backslash\Bigr)^2 \\ \$\$$$

Команды, явно указывающие размер ограничителей, не обязаны, в отличие от команд $\backslash\left$ и $\backslash\right$, появляться парами: можно без всяких ухищрений написать $\backslash\biggl($ и при этом никак не упомянуть о парной скобке.

К сожалению, команды для явного указания размера ограничителя имеют, при использовании их в \TeX'e , одну неприятную особенность: если «основной шрифт» документа крупнее, чем кегль 10 (иными словами, если указаны стилевые опции $11pt$ или $12pt$ — см. с. 141 и ниже), то может случиться так, что скобка, размер которой задан, например, командой $\backslash\bigl$, имеет точно такой же размер, как и скобка «в чистом виде». Поэтому при необходимости явного указания размеров ограничителей конкретную команду для выбора размера приходится иной раз подбирать экспериментально.

- Если вы пользуетесь \LaTeX'ом 2 ε , то у вас есть возможность избежать этой неприятности, подключив AMS-\TeX'овские стилевые пакеты, о которых пойдет речь в приложении Е.

2.7. Перечеркнутые символы

Чтобы получить в математической формуле изображение перечеркнутого символа, надо перед командой, генерирующей этот символ, поставить команду $\backslash\cancel$. Пример:

Множество $\{x : x \not\not x\}$ существовать не может. В этом состоит парадокс Рассела.

Кстати, для получения знака \notin лучше писать \backslashnotin , а не $\backslashnot\in$, — при этом знак получится более красивым.

Множество $\{\{x : x \not\in x\}\}$ существовать не может. В этом состоит парадокс Рассела.

2.8. Надстрочные знаки

Часто требуется поставить дополнительный значок над буквой или фрагментом формулы: черточку, «крышку», и т. п. В \TeX 'е для этих целей есть специальные команды.

Во-первых, можно поставить горизонтальную черту над любым фрагментом формулы с помощью команды $\overline{...}$, как в следующем примере:

Часто используется обозначение

$$\overline{a_n a_{n-1} \dots a_1 a_0} = 10^n a_n + \dots + a_0.$$

Особенно часто так пишут в научно-популярных книгах.

Часто используется обозначение

$\$$

$\overline{a_n a_{n-1} \dots a_1 a_0} = 10^n a_n + \dots + a_0.$

$\$$

Особенно часто так пишут в научно-популярных книгах.

Для постановки других значков над буквами в формулах предусмотрены команды, перечисленные в следующей таблице, в которой, для примера, эти значки ставятся над буквой *a*:

\hat{a}	\check{a}
\tilde{a}	\acute{a}
\grave{a}	\dot{a}
\ddot{a}	\breve{a}
\bar{a}	\vec{a}

Между прочим, команда \bar{a} ставит не совсем такую же черточку, как $\overline{...}$.

Если поставить значок над буквой *i* или *j*, чтобы сохранилась и точка над буквой, то это будет некрасиво. Поэтому значки следует ставить не прямо над этими буквами, а над символами *i* и *j* (см. таблицу на с. 50):

Писать \hat{i} некрасиво; лучше писать так: \hat{i} .

Писать \tilde{i} некрасиво; лучше писать так: $\tilde{\imath}$.

Надстрочные знаки, перечисленные в таблице, можно ставить только над одиночными буквами: если сказать $\hat{a+b}$, то получится некрасивая формула $\hat{a} + b$. \TeX предоставляет возможность поставить «крышку» подходящего размера над целым фрагментом формулы с помощью команды $\widehat{...}$:

Тождество $\widehat{f * g} = \widehat{f} \cdot \widehat{g}$ означает, что преобразование Фурье переводит свертку в произведение.

Тождество $\widehat{\widehat{f} \cdot \widehat{g}} = f \cdot g$ означает, что преобразование Фурье переводит произведение в свертку.

Аналогичным образом можно поставить «волну» над фрагментом формулы с помощью команды `\widetilde`. В отличие от горизонтальной черты, генерируемой командой `\overline`, знаки, генерируемые командами `\widehat` и `\widetilde`, не могут быть сколь угодно широкими (максимально возможная ширина — в примере выше).

Кроме того существует команда `\overrightarrow`, предназначенная для постановки стрелки над формулой:

Рассмотрим вектор \vec{AB} .

Рассмотрим вектор
 \overrightarrow{AB} .

Аналогичная ей команда `\overleftarrow` ставит над формулой стрелку, направленную влево, а не вправо.

Остальные команды для постановки акцентов в формулах не имеют «широких» вариантов.

Формулы типа `\hat{\hat{A}}`, в которых акцент ставится над буквой, уже имеющей акцент, смотрятся при наборе в L^AT_EX'е неудачно. Если вы пользуетесь L^AT_EX'ом 2_ε, то можно избавиться от этого недостатка с помощью *AMS-L^AT_EX'a*. (приложение E).

TeX позволяет поставить надстрочные знаки над буквами не только в математической формуле, но и в обычном тексте (в этом случае такие знаки обычно называют «диакритическими»), но команды для постановки этих знаков совершенно другие. Об этом — на с. 92.

2.9. Альтернативные обозначения для математических формул

Наряду со стандартными TeX'овскими обозначениями для математических формул, L^AT_EX предоставляет альтернативные обозначения. Именно, внутритечстовую формулу, которая в стандартных обозначениях ограничивается одним знаком доллара в начале и одним в конце, можно вместо этого заключить в знаки `\(`` (в начале) и `\)`` (в конце). Другой вариант обозначений для внутритечстовой формулы, предоставляемый L^AT_EX'ом, — написать `\begin{math}` в начале формулы и `\end{math}` в конце (иными словами, внутритечстовая формула может быть оформлена как окружение с именем `math`:

Выключную формулу \LaTeX позволяет окружить с обеих сторон не только парами знаков доллара, как предусмотрено стандартом, но знаками $\backslash[$ (в начале) и $\backslash]$ (в конце). Кроме того, можно оформить выключную формулу как окружение с именем `displaymath`. В одном и том же файле можно использовать как стандартные, так и \LaTeX 'овские обозначения для формул.

Эти альтернативные обозначения полностью эквивалентны стандартным (со знаками доллара), за одним важным исключением: если выключные формулы обозначаются \LaTeX 'овскими, а не \TeX 'овскими обозначениями, то можно сделать так, что выключные формулы будут не центрированы, а прижаты влево (см. с. 143).

Создатель \LaTeX 'а Лесли Лэмпорт надеялся, что его обозначения для формул будут удобнее стандартных, поскольку в них различаются символы, «открывающие» и «закрывающие» формулу, что позволяет легче находить ошибки в исходном тексте. По мнению автора, эти надежды отчасти оправдались для коротких обозначений $\backslash($, $\backslash)$, $\backslash[$ и $\backslash]$, в то время как более длинные обозначения, использующие `\begin{` и `\end`, оказались слишком громоздкими. В пользу стандартных обозначений говорит то обстоятельство, что при переводе файла из \LaTeX 'а в plain \TeX или \AMSTEX не приходится заменять на доллар каждый знак $\backslash($ и т. д.

3. Одно над другим

В этом разделе речь пойдет о тех случаях, когда в формуле необходимо поместить один символ над другим. В разд. 1.2 уже шла речь о частном случае этой проблемы: постановке «пределов» у знака суммы, интеграла или чего-нибудь еще в этом роде. Сейчас мы рассмотрим общий случай.

3.1. Простейшие случаи

Для начала рассмотрим такие возможности расположения одной части формулы над другой:

- 1) Верхняя часть формулы расположена немного выше строки, нижняя — немного ниже (как в дроби, создаваемой командой `\frac`, но без дробной черты).
- 2) Нижняя часть формулы расположена вровень с остальным текстом, верхняя — над ним.

- 3) Над или под фрагментом формулы проведена горизонтальная фигурная скобка, а над или под этой скобкой расположен другой фрагмент формулы.

Разберем эти варианты последовательно.

Чтобы расположить части формулы по варианту 1, удобно воспользоваться $\text{\TeX}'$ овской командой `\atop`:

Раньше вместо Γ_{ij}^k писали $\left\{ \begin{smallmatrix} ij \\ k \end{smallmatrix} \right\}$.

Раньше вместо Γ_{ij} писали $\left[ij\atop k\right]$.

В данном случае мы воспользовались еще командами `\left` и `\right` для постановки фигурных скобок необходимого размера.

Часто встречающийся случай, когда выражения должны быть расположены так, как это делает команда `\atop` — это «биномиальные коэффициенты»; их можно было бы набирать как в вышеприведенном примере, с помощью `\left(`, `\atop` и `\right)`, но быстрее воспользоваться готовой командой `\choose`, которую предоставляет нам ТЕХ:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

\$\$

$$\{n \choose k\} = \frac{n!}{k!(n-k)!}$$

\$\$

Обратите внимание на фигурные скобки, в которые мы заключили выражение $n \choose k$: команда \choose помещает сверху часть формулы от открывающей фигурной скобки до \choose , а снизу — часть формулы от \choose до закрывающей фигурной скобки. Если бы этих фигурных скобок не было, вниз пошла бы и вся дробь $\frac{n!}{k!(n-k)!}$ вместе со знаком равенства.

Команда `\atop` определяет, что пойдет вверх, а что — вниз, по тем же правилам, что и `\choose`. В примере выше с `\atop` мы обошлись без фигурных скобок, поскольку в математической формуле их функцию исполняют также команды `\left` и `\right`.

Рассмотрим второй случай, когда нижняя часть формулы должна остаться на уровне строки. Чтобы добиться этого эффекта, используется L^AT_EX'овская команда `\stackrel`. У этой команды два аргумента: первый — то, что будет над строкой, второй — то, что останется в строке:

$$A \xrightarrow{f} B$$

\$A\stackrel{f}{\longrightarrow}B\$

Наконец, чтобы нарисовать горизонтальную фигурную скобку под выражением (а под этой скобкой еще, возможно, и сделать подпись), надо воспользоваться командой `\underbrace`. Аргумент этой команды —

тот фрагмент формулы, под которым надо провести скобку; подпись под скобкой, если она нужна, оформляется как нижний индекс. Например, такая формула

$$\underbrace{1 + 3 + 5 + 7 + \cdots + 2n - 1}_n \text{ слагаемых} = n^2$$

получается следующим образом:

2

$\underbrace{1+3+5+7+\dots+}_{\text{сумма } n \text{ слагаемых}} = n^2$

\$\$

Горизонтальная фигурная скобка над фрагментом формулы генерируется командой `\overbrace`, надпись над ней оформляется как верхний индекс. В одной формуле могут присутствовать горизонтальные фигурные скобки как над, так и под фрагментом формулы:

$$\begin{array}{c}
 \overbrace{a+b+\cdots+z+1+\cdots+10}^{36} \\
 26
 \end{array}
 \quad
 \begin{aligned}
 & \text{\$\$} \\
 & \backslash overbrace{ \backslash underbrace{ }_{\{26\}+1+} } \\
 & \quad a+b+\cdots+z \\
 & \quad \} \\
 & \quad \backslash cdots +10 \}^{\{36\}} \\
 & \text{\$\$}
 \end{aligned}$$

Разумеется, совсем не обязательно записывать формулу с такими отступами и пробелами, как в нашем примере; мы пытались расположить текст так, чтобы яснее была его структура, благо пробелы в формулах \TeX все равно игнорирует.

В нашем примере нижняя горизонтальная скобка была расположена целиком внутри верхней горизонтальной скобки. Можно сделать и так, чтобы верхняя и нижняя горизонтальные скобки не содержали одна другую, а перекрывались, но для этого нужны дополнительные хитрости. См. с. 80.

Рассмотренные приемы не исчерпывают всех случаев, когда требуется расположить одну часть формулы над другой. Например, желательно, чтобы элементы матрицы, расположенные в одном столбце, и на бумаге были расположены точно один под другим. Добиться этого с помощью `\atop` практически невозможно. В следующем разделе мы опишем, как грамотно набирать матрицы, системы уравнений и т. п.

3.2. Набор матриц

Чтобы набрать с помощью \LaTeX 'а матрицу, надо воспользоваться окружением `array`. Прежде чем описывать, как это окружение работает, разберем такой пример:

```


$$\begin{array}{cccc}
a_{11} & a_{12} & \dots & a_{1n} \\
a_{21} & a_{22} & \dots & a_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
a_{n1} & a_{n2} & \dots & a_{nn}
\end{array}$$

$$

```

Посмотрим, как устроен исходный текст, давший на печати эту матрицу. Всякая матрица состоит из *строк* и *столбцов*; строки матрицы разделяются с помощью команды `\backslash` (последнюю строку заканчивать командой `\backslash` не надо), а элементы внутри одной строки, относящиеся к разным столбцам, отделяются друг от друга с помощью символа `&`. Далее, после `\begin{array}`, открывающего окружение, должна следовать (в фигурных скобках, поскольку это параметр окружения `array`) так называемая *преамбула* матрицы, описывающая, сколько и каких столбцов должно быть в матрице. В нашем случае преамбула представляет собой четыре буквы `cccc`. Это значит, что в матрице 4 столбца (по букве на столбец), и что содержимое каждого из этих столбцов должно быть расположено по центру столбца (`c` — от слова “centered”). Кроме `c`, в преамбуле может стоять буква `l`, означающая, что соответствующий столбец будет выровнен по левому краю (`left`), или `r`, означающая, что столбец будет выровнен по правому краю (`right`).

Мы использовали здесь многоточие на строке, генерируемое командой `\ldots`, вертикальное многоточие, генерируемое командой `\vdots`, и диагональное многоточие, генерируемое командой `\ddots`. Эти команды можно использовать не только внутри окружения `array`, но и в любом месте в математических формулах.

Нашей матрице не хватает еще скобок; чтобы их создать, надо написать `\left(` перед `\begin{array}` и `\right)` после `\end{array}` (см. разд. 2.6).

Внутри окружения `array` разбиение текста на строки роли не играет: бывает, что на протяжении нескольких строк приходится тянуть текст, который превратится в одну строку на печати, а иногда в одной строке исходного текста помещается несколько строк матрицы. Где должна кончаться строка на печати, определяется командой `\backslash`.

Окружение `array` можно, разумеется, использовать не только для матриц в математическом смысле этого слова: это окружение просто создает таблицы, состоящие из строк и столбцов. Вот, например, как можно напечатать треугольник Паскаля:

		1		1	
		1	2	1	
	1	3	3	1	
1	4	6	4	1	
1	5	10	10	5	1

Исходный текст для него выглядит так:

Как видите, если какая-то графа в нашей таблице должна быть пуста, то между (или перед, если эта графа — первая в своей строке) соответствующими знаками & нужно просто ничего не писать (или оставить сколько угодно пробелов). Если после того, что вы написали в строке, до конца строки идут только пустые графы, то можно не дописывать до конца значки &, а сразу написать \\.

Разберем еще один пример, типичный при работе в L^AT_EX'е 2.09: верстку системы уравнений с помощью окружения `array`. Посмотрите, как получается такая система уравнений:

```


$$\left\{ \begin{array}{l} x^2 + y^2 = 7 \\ x + y = 3. \end{array} \right.$$


```

Мы отвели по одному столбцу на левую часть каждого уравнения, на знак равенства и на правую часть. При этом мы попросили, чтобы левые

части уравнений были выровнены по правому краю (отсюда `r` в преамбуле), правые части выровнены по левому краю (`l` в преамбуле), а знак равенства располагался по центру своей колонки (поэтому вторая буква в преамбуле — буква `c`). Для создания фигурной скобки, охватывающей всю систему, мы воспользовались командами `\left` и `\right`, причем при команде `\right` стоит «пустой ограничитель» — точка (см. разд. 2.6).

Обратите внимание, что пробелы (отбивки) до и после знака равенства получаются больше, чем это допускается типографскими правилами. Если вы работаете в \LaTeX е версии 2.09, то вам придется либо с этим смириться, либо проставить отрицательные пробелы `\!` с двух сторон знака равенства (см. с. 76 по поводу отрицательного пробела). При работе в версии \LaTeX 2 ε таким образом набирать системы уравнений нет никакой необходимости, поскольку для этого есть иные, более совершенные средства (см. приложение Е, описание окружения `aligned`).

Если необходимо, чтобы отдельные уравнения в системе были пронумерованы, можно воспользоваться окружением `eqnarray`. Оно работает так же, как окружение `array` с преамбулой `rcl` в вышеприведенном примере, но при этом у каждого уравнения автоматически печатается его номер (подобно тому, как автоматически печатается номер у выключной формулы, созданной с помощью окружения `equation` — см. разд. 2.1). Если пометить какое-либо уравнение с помощью команды `\label`, то в дальнейшем можно на него ссылаться с помощью команды `\ref` (тогда автоматически напечатается номер уравнения) или `\pageref` (тогда автоматически напечатается номер страницы, на которую попало это уравнение). По поводу `\ref` и `\pageref` см. разд. I.2.11. Итак, пример:

	$2 \times 3 = 6$	$2 + 3 = 5$
	(2)	(3)

На с. 70 приведено глупое уравнение 3.

```

\begin{eqnarray}
2\times3&=&6\\
2+3&=&5\label{silly}\\
\end{eqnarray}
На с.~\pageref{silly} приведено глупое уравнение~\ref{silly}.

```

Обратите внимание, что фигурной скобки, охватывающей систему уравнений, окружение `eqnarray` не создает. В этом примере символ `-` между `«с.»` и `\pageref` поставлен, чтобы слово `«с.»` и номер страницы не попали на разные строки (см. с. 89); для аналогичных целей мы использовали этот символ и вторично.

При использовании окружения `\eqnarray` не надо писать знаков `$$` (подобно тому, как не надо их писать при использовании окружением `\equation`).

Если вы хотите нумеровать не все уравнения, надо уравнения, которые вы нумеровать *не* будете, пометить командой `\nonumber` (непосредственно перед `\backslash`):

$$\begin{aligned} \int_{-\infty}^{\infty} e^{-x^2} dx &= \sqrt{\pi} \\ \sqrt{576} &= 24 \quad (4) \end{aligned} \quad \begin{aligned} &\begin{aligned} &\backslash\begin{aligned} &\text{begin}\{eqnarray} \\ &\text{int}_f\{-\infty\}^{\infty} \\ &e^{-x^2}dx \& = \& \\ &\text{sqrt}\{\pi\}\text{nonumber}\\ &\text{sqrt}\{576\} \& = \& 24 \\ &\text{end}\{eqnarray} \end{aligned} \end{aligned}$$

Наконец, если вы вообще не хотите нумеровать уравнения, то можно воспользоваться «вариантом со звездочкой» — окружением `\eqnarray*` вместо `\eqnarray`.

- ε Пользователю $\text{\LaTeX}'$ а 2ε опять-таки лучше окружением `\eqnarray` не пользоваться, а подключить стилевой пакет `amsmath` и пользоваться, например, окружением `align`.

Окружение `array` можно использовать не только в выключочных, но и во внутритеекстовых формулах, хотя результат при этом обычно выглядит некрасиво. Окружения `\eqnarray` и `\eqnarray*` создают только выключные формулы.

Давайте теперь подумаем, как с помощью окружения `array` можно набирать так называемые «коммутативные диаграммы» такого, например, вида:

$$\begin{array}{ccccccc} 0 & \longrightarrow & E' & \xrightarrow{f} & E & \xrightarrow{g} & E'' \longrightarrow 0 \\ & & \downarrow p & & \downarrow q & & \downarrow r \\ 0 & \longrightarrow & F' & \xrightarrow{f} & F & \xrightarrow{g} & F'' \longrightarrow 0 \end{array}$$

- ε Если вы пользуетесь $\text{\LaTeX}'$ ом 2ε, то можете при первом чтении пропустить следующее ниже описание ухищрений, необходимых для печати этой диаграммы. Вместо этого прочтите следующий раздел 3.3, где описано, как удобно набирать коммутативные диаграммы с помощью окружения `CD`.

Разумно реализовать эту диаграмму как таблицу с тремя строками и девятью столбцами (по столбцу на каждую горизонтальную стрелку и на каждый нуль или букву). Как создать буквы над горизонтальными стрелками, мы тоже уже говорили: с помощью `\stackrel` (см. с. 66).

Осталось понять, как получить буквы справа от вертикальных стрелок. Для экономии попробуем сначала нарисовать диаграмму с *одной* вертикальной стрелкой. Вот первая попытка:

E	\$
$\downarrow q$	$\begin{array}{c} \\ \backslash \downarrow \\ \end{array}$
F	F

\$\$

```
\begin{array}{c}
E \\ \downarrow q \\ F
\end{array}
```

\$\$

Вроде бы получилось, но не совсем то, что нужно: стрелка расположена не по центру. Не улучшит положения, если в преамбуле написать `g` вместо `c`; если написать `l`, то будет чуть лучше, но стрелка все равно будет не в центре. Чтобы всего этого избежать, надо вместо `q` написать `\leftarrow q`: после этого столбцы нашей таблицы будут отцентрированы так, словно во второй строке есть только стрелка (команда `\downarrow`), а буква `q` при этом центрировании принята во внимание не будет (но будет напечатана)! В результате получится именно то, чего мы и хотели.

Для интересующихся объясним механизм действия этого приема. В процессе верстки текста \TeX учитывает только, сколько места надо оставить на каждую букву, но не учитывает, как именно эта буква будет выглядеть на печати и сколько места она реально будет занимать. В обычных условиях на каждый символ оставляется именно столько места, сколько он занимает на самом деле, однако в \TeX 'е предусмотрены и специальные команды, позволяющие отвести тексту меньше или больше места, чем он займет фактически. В частности, команда `\leftarrow q` печатает формулу, являющуюся ее аргументом, но при этом сообщает \TeX 'у, что по горизонтали эта формула не занимает места вообще. Стало быть, с точки зрения \TeX 'а ширина элемента, стоящего во второй строке нашей таблицы, определяется только шириной стрелки, и при центрировании текста располагается так, чтобы именно стрелка была на равном расстоянии от краев, сколь бы длинна на самом деле ни была формула — аргумент `\leftarrow q`. Создатель \TeX 'а Дональд Кнут назвал такого рода приемы работы с \TeX 'ом «грязными трюками» (dirty tricks). Впрочем, при написании таких больших и сложных \TeX 'овских макропакетов, как \LaTeX , используются трюки и похлеще.

Теперь можно сделать и всю коммутативную диаграмму целиком. Держитесь:

```
$$
\begin{array}{ccccccccc}
& & & & & & & & \\
0 & \longrightarrow & E' & & & & & & \\
\stackrel{f}{\longrightarrow} & & \longrightarrow & E & & & & & \\
\stackrel{g}{\longrightarrow} & & & & \longrightarrow & & & &
\end{array}
```

```

E'' & \longrightarrow & O\\
&&\downarrow\lefteqn{p}\&&\downarrow\lefteqn{q}\\
\lefteqn{q}\&&\downarrow\lefteqn{r}\\
O\&\longrightarrow & F' &
\stackrel{f}{\longrightarrow} & F &
\stackrel{g}{\longrightarrow} & F''\\
& \longrightarrow & O\\
\end{array}
$$

```

Как видите, писаницы очень много. Если вы реально нуждаетесь в подобных вещах, посмотрите уже сейчас начало гл. VII: там объяснено, как можно значительно сократить длину подобных записей.

Возможности окружения `array` далеко не исчерпываются тем, что было рассказано в этом разделе, однако сказанного в большинстве случаев хватает; полное описание возможностей этого окружения будет дано позже, в гл. VI.

3.3. Набор коммутативных диаграмм в $\text{\LaTeX}'e$ 2 _{ε}

Как и было обещано, расскажем теперь, как можно резко упростить набор коммутативных диаграмм в $\text{\LaTeX}'e$ 2 _{ε} . Для этого надо подключить стилевой пакет `amscd`, то есть либо вставить после команды `\documentclass` строку

```
\usepackage{amscd}
```

либо добавить слово `amscd` через запятую в аргумент одной из уже существующих команд `\usepackage`. Пусть это сделано. Тогда коммутативная диаграмма оформляется в виде окружения `CD`. Читателью, знакомому с \AMSTEX 'ом, дальнейшее можно объяснить одной фразой: между `\begin{CD}` и `\end{CD}` надо поместить в точности тот же текст, что в \AMSTEX 'е пишут в аналогичном случае между `\CD` и `\endCD` (см. [6, гл. 19]). Для всех остальных удобнее пояснить правила набора коммутативных диаграмм на примере. Вот какой исходный текст породит диаграмму со с. 71:

```

$$
\begin{CD}
@>>> E' @>f>> E @>g>> E'' @>>> O\\
@. @VVpV @VVqV @VVrV @.\\
@>>> F' @>f>> F @>g>> F'' @>>> O\\
\end{CD}
$$

```

Первая строка в этой записи соответствует верхней строке диаграммы. Стрелка, направленная слева направо, задается конструкцией $\text{\texttt{>>>}}$ (а стрелка справа налево — конструкцией $\text{\texttt{<<<}}$); если над стрелкой надо поставить какую-то надпись (например, просто букву), то нужно ее разместить между первым и вторым знаками неравенства; чтобы надпись получилась под стрелкой, надо ее разместить между вторым и третьим знаками неравенства.

Вторая строка задает вертикальные стрелки. Конструкция $\text{\texttt{&V\&V}}$ задает стрелку, направленную вниз; если справа от стрелки нужна надпись, то ее надо разместить между второй и третьей буквой V (чтобы надпись оказалась слева от стрелки, она должна быть, естественно, между первой и второй буквой V). Вертикальная стрелка, направленная вверх, задается конструкцией $\text{\texttt{&A\&A}}$ (буква A — максимальное приближение к устремленной вверх стреле); справа и слева от нее также можно сделать надпись (аналогичным образом).

Конструкция $\text{\texttt{&}}$ задает «пустую» стрелку (в нашем случае — между двумя нулями); она необходима, чтобы $\text{\LaTeX} 2\varepsilon$ не сбился со счета, выясняя, в какие колонки ставить вертикальные стрелки.

Теперь опишем работу окружения CD более аккуратно. Каждую коммутативную диаграмму окружение CD рассматривает как таблицу, состоящую из перемежающихся «горизонтальных» и «вертикальных» строк. Каждая «горизонтальная» строка состоит из формул, перемежающихся горизонтальными стрелками. Во всех горизонтальных строках должно быть одинаковое количество формул. Если некоторые из мест, предназначенных для формул, должны остаться пустыми, то на этом месте надо оставить пробел или, если вам так приятнее, написать $\{\}$. Между каждой парой формул должна быть стрелка. Если какие-то из этих стрелок не нужны, на их месте надо поставить $\text{\texttt{&}}$. (пустую стрелку).

Каждая «вертикальная» строка состоит из вертикальных стрелок. Их должно быть столько же, сколько формул в любой из горизонтальных строк. Если какие-то из вертикальных стрелок не нужны, на их месте надо поставить $\text{\texttt{&}}$. (пустую стрелку).

Если надпись при стрелке, направленной вниз (и задаваемой, стало быть, конструкцией $\text{\texttt{&V\&V}}$), сама содержит букву V, то нужно ее (надпись) взять в фигурные скобки — иначе \TeX не сможет понять, какая из букв V относится к надписи, а какая — к обозначению стрелки. Аналогичные меры надо принять, если надпись при стрелке, направленной вверх, содержит букву A (а также, естественно, если надпись при горизонтальной стрелке содержит знак $>$ или $<$, хотя ввиду математического смысла таких надписей последнее маловероятно).

Наряду со стрелками, в коммутативных диаграммах встречаются горизонтальные и вертикальные «растянутые знаки равенства»:

$$\begin{array}{ccccc}
 A & \xlongequal{\hspace{1cm}} & B & \longrightarrow & C \\
 v_1 \downarrow & & f \uparrow & & \parallel \\
 D & \xleftarrow{g} & E & \xleftarrow{\hspace{1cm}} & F
 \end{array}
 \quad \begin{aligned}
 & \text{\$\$} \\
 & \begin{aligned}
 & \backslash\begin{CD} \\
 & A @= B @>>> C \\
 & @V{v_1}VV @A{f}AA @| \\
 & D @<< g < E @<<< F
 \end{aligned} \\
 & \backslash\end{CD} \\
 & \text{\$\$}
 \end{aligned}$$

Как видно из этого примера, такие знаки задаются конструкциями `@=` (горизонтальный) и `@|` (вертикальный). Обратите также внимание, как мы защитили фигурными скобками символ `V` в надписи к левой вертикальной стрелке.

Конструкция `\pretend... \haswidth` из книги [6] в $\text{\LaTeX}'e 2_{\epsilon}$ не поддерживается.

Остается только еще раз подчеркнуть, что все возможности, описанные в этом разделе, доступны только при использовании $\text{\LaTeX}'e 2_{\epsilon}$. В $\text{\LaTeX}'e 2.09$ придется набирать коммутативные диаграммы с помощью приемов, описанных в конце предыдущего раздела.

3.4. Переносы в выключных формулах: $\text{\LaTeX} 2.09$

Как уже отмечалось, переносов в выключных формулах \TeX автоматически не делает, поэтому при необходимости приходится делать такие переносы вручную. Для этого удобно использовать уже знакомые нам окружения `array`, `eqnarray` или `eqnarray*`. В самом деле, всякую формулу из нескольких строк можно рассматривать как матрицу с одним столбцом:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots
 \quad \begin{aligned}
 & \text{\$\$} \\
 & \begin{aligned}
 & \backslash\begin{array}{l} \\
 & e^x=1+x+\frac{x^2}{2!}\backslash \\
 & \qqquad {}+\frac{x^3}{3!}+\cdots \\
 & \backslash\end{array} \\
 & \text{\$\$}
 \end{aligned}
 \end{aligned}$$

Команда `\qqquad` делает в тексте или формуле отступ длины 2em («два пробела»: см. разд. I.1 по поводу единицы длины em и 4.1 по поводу пробелов внутри формул). Если бы этой команды не было, то части формулы на двух строках начинались бы точно одна под другой, что менее понятно и считается неграмотным набором.

Надо еще объяснить, зачем мы поставили {} перед знаком «плюс» во второй строке. Сделано это затем, чтобы TeX сделал правильный интервал между плюсом и $\frac{x^3}{3!}$: скобки {} ограничивают «пустую подформулу», первый из плюсов во второй строке оказывается между двумя формулами, что и приводит к пробелу надлежащего размера. Мы будем еще говорить об этих вещах в разд. 4.4.

Вместо окружения `array` можно было бы использовать `eqnarray*` или `eqnarray`: при пользовании последним окружением формула заодно получит и номер. Разумеется, при этом перед \\", завершающим обрвянную строку, надо поставить команду `\nonumber`, если вы не хотите, чтобы и обрубок формулы был пронумерован.

Если вы пользуетесь L^AT_EX'ом 2_E, то для печати многострочных выключных формул лучше всего пользоваться возможностями, описанными в разд. 4 приложения Е (см. в особенности окружение `multline`).

4. Тонкая настройка

В этом разделе мы рассмотрим некоторые более изысканные вопросы, связанные с набором математических формул.

4.1. Пробелы вручную

Бывают случаи, когда промежутки между символами в формулах, выбранные TeX'ом автоматически, выглядят неудачно. В этом случае в формулу можно включить команды, задающие промежутки в явном виде. Вот основные из них:

<code>\quad</code>	Пробел в 1em
<code>\quad\quad</code>	Пробел в 2em
<code>\,</code>	«Тонкий пробел», или тонкая шпация
<code>\:</code>	«Средний пробел»
<code>\;</code>	«Толстый пробел»
<code>\!</code>	«Отрицательный тонкий пробел»

Команда `\!` из этой таблицы *уменьшает* промежуток на столько же, на сколько команда `\,` его увеличивает.

В следующем примере собраны типичные случаи, когда в этих командах возникает нужда.

Пробелы надо корректировать в таких формулах, как $\int f(x) dx$, $\iint f dxdy$ или $\sqrt{3} x$.

Пробелы надо корректировать в таких формулах, как $\$\\int f(x)\\,dx$, $\$\\int\\!\\!\\!\\int f\\,dxdy$$ или $\$\\sqrt{3}\\,x$.$$

Команда `\quad` полезна для отделения текста, входящего в формулу, от собственно формулы (см. с. 58). Для этих же целей можно использовать команду `\quad`, делающую пробел размером 1em.

4.2. Размер символов в формулах

В большинстве случаев вам не приходится задумываться о том, какой размер будут иметь символы в формуле: как вы, вероятно, уже заметили, TeX автоматически выбирает более мелкий шрифт для степеней, индексов, числителей и знаменателей дробей, созданных командой `\frac`, и т. п. Бывают, однако, случаи, когда в этот процесс автоматического выбора размара приходится вмешаться. Сейчас мы вкратце опишем, как TeX выбирает размеры символов в формулах и как можно на него при этом влиять.

При наборе формулы TeX в каждый момент руководствуется одним из следующих «стилей» (эти стили не имеют ничего общего со стилями оформления документа, о которых говорилось в разд. I.2.4):

<code>displaystyle</code>	«выключной» стиль
<code>textstyle</code>	«текстовый» стиль
<code>scriptstyle</code>	стиль для индексов
<code>scriptscriptstyle</code>	стиль для индексов к индексам.

«Выключной» и «текстовый» стили используют одинаковые шрифты, но формулы в текстовом стиле выглядят чуть «поскромнее» (например, в выключном стиле верхние индексы поднимаются повыше, а нижние — пониже, чем в текстовом; в текстовом стиле «пределы» операций записываются не сверху, а сбоку — ср. разд. 1.2). В стиле для индексов используются более мелкие шрифты, чем в выключном или текстовом (а в стиле для индексов к индексам — еще более мелкие). Выбираются стили набора формул следующим образом: выключная формула начинает набираться в выключном стиле, внутритекстовая — в текстовом стиле; далее, если в момент действия какого-то из стилей встретится команда `\frac` (или `\atop`), то для набора числителя и знаменателя TeX переключается на следующий по порядку стиль из вышеприведенной таблицы; если в момент действия выключного или текстового стиля

встретится верхний или нижний индекс (показатель степени мы также рассматриваем как верхний индекс — в математическое содержание формул TeX не вникает), то этот индекс начинает набираться стилем для индексов; если индекс встретится в момент действия стиля для индексов или индексов к индексам, то набираться он будет в стиле «индексы к индексам». Например, при наборе формулы

$$x + \frac{y^2 + 3 - z^{x^7 - y^7}}{1 + \cos^2 x}$$

TeX использует выключной стиль при наборе $x+$, текстовый стиль при наборе $y^2+3-z^{x^7-y^7}$ и $1+\cos^2 x$, стиль для индексов при наборе x^7-y^7 , и стиль для индексов к индексам при наборе семерок в показателях степени. Стиля **scriptscriptstyle** и дальнейших не предусмотрено, так что индексы третьего и более высоких порядков набираются теми же шрифтами, что и индексы второго порядка (расстраиваться по этому поводу не надо, поскольку эти шрифты и так мелкие).

Если вы хотите изменить стиль набора формулы, можно в явном виде указать его с помощью TeX'овской команды, имя которой совпадает с английским названием этого стиля (`\displaystyle ... \scriptstyle`). Вот типичный пример, когда это может понадобиться. Предположим, в вашем тексте встречаются «цепные дроби»:

$$\frac{7}{25} = \cfrac{1}{3 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{3}}}}$$

Наивная попытка набрать эту формулу выглядит так:

$$\frac{7}{25} = \cfrac{1}{3 + \cfrac{1}{1 + \cfrac{1}{1 + \frac{1}{3}}}}$$

\$\$

```
\frac{7}{25}=
\frac{1}{3+\frac{1}{1+\frac{1}{1+\frac{1}{3}}}}
```

\$\$

Результат несколько несуразен, но формально полностью согласуется с правилами TeX'a: вся формула, коль скоро она выключная, набирается в выключном стиле, стало быть числитель и первый из знаменателей будут уже в текстовом стиле, следующий знаменатель — как индексы, следующий — как индексы к индексам, и т. д. Надо было набирать так:

```
$$
\frac{7}{25}=
\frac{1}{\displaystyle 3+\frac{1}{\displaystyle 1+\frac{1}{\displaystyle 1+\frac{1}{\displaystyle 3}}}}
$$
```

Каждая из трех команд `\displaystyle` необходима для того, чтобы каждая из последующих дробей набиралась в выключном стиле, невзирая на то, что она сама стоит в знаменателе.

- ε В приложении Е будет рассказано о более удобном способе набора цепных дробей, предоставляемом *ЛМС-Л^EX*'ом.

4.3. Фантомы и прочее

В разд. 3.2 мы столкнулись с командой `\lefteqn`, позволяющей напечатать фрагмент формулы и при этом сообщить *TeX*'у, что отдельного места (по горизонтали) на этот фрагмент отводить не надо. Иногда бывает полезно сделать обратное: включить в формулу символ, который сам не печатается, но место занимает. Вот пример такой ситуации.

Команда `\sqrt` автоматически выбирает размер знака радикала таким образом, чтобы он точно соответствовал высоте подкоренного выражения, и это очень хорошо. Иногда, однако, такой автоматический выбор приводит к не очень удачным результатам:

В формуле $\sqrt{a} + \sqrt{d}$ два знака радикала имеют разные размеры.

В формуле $\sqrt{a} + \sqrt{d}$ два знака радикала имеют разные размеры.

Дело тут, конечно, в том, что буквы *a* и *d* имеют разную высоту. Чтобы сделать знаки радикала одинаковыми, *TeX* надо обмануть: добавить в подкоренные выражения по символу, который чуть выше, чем *a* или *d*, чтобы подкоренные выражения оказались одной высоты. Этот символ, естественно, не должен печататься и не должен занимать места по горизонтали (лишние пробелы под корнем нам тоже ни к чему). Такой невидимый символ генерируется *TeX*'овской командой `\mathstrut`:

В формуле $\sqrt{a} + \sqrt{d}$ оба знака радикала имеют одинаковые размеры.

В формуле $\sqrt{\mathstrut a} + \sqrt{\mathstrut d}$ оба знака радикала имеют одинаковые размеры.

Точнее говоря, `\mathstrut` — это невидимый символ, равный по высоте скобке (и не имеющий ширины.

Невидимый символ, создаваемый командой `\mathstrut`, является частным случаем TeX'овской конструкции «фантома». Именно, если в формуле вы напишете

``

то результат будет такой же, как если бы эта самая «какая-то формула» была сначала напечатана по всем правилам TeX'a, а затем аккуратно стерта с бумаги (но следов ластика вы не увидите). Пример:

Все мы знаем, что знак радикала выглядит так: $\sqrt{}$.

Все мы знаем, что знак радикала выглядит так: `-$\sqrt{}$`.

Кроме того, можно создать «вертикальный фантом» формулы (по вертикали будет оставлено столько же места, сколько занимала бы формула, по горизонтали вертикальный фантом места не занимает). Создается вертикальный фантом командой `\vphantom`. В частности, команда `\mathstrut` — это сокращение для `\vphantom{()}`. Возможны, наконец, и горизонтальные фантомы, занимающие по горизонтали столько же места, сколько заняла бы формула, и не занимающие места по вертикали. Создаются они командой `\hphantom`:

На пустое место можно вписать формулу вручную.

На пустое место можно вписать формулу вручную.

Для полноты картины скажем об еще одной экзотической команде, называемой `\smash`. Подобно команде `\lefteqn`, она печатает символ, но при этом говорит TeX'у, что он не занимает места по вертикали. С помощью этой команды (а так же с помощью `\lefteqn`) можно накладывать в формулах один символ на другой. Вот пример с `\lefteqn`:

$$\overbrace{1+2+3+4}^{}$$

```
$$
\lefteqn{\overbrace{
\phantom{1+2+3}}
^{1+\underbrace{2+3+4}}}
$$
```

Поясним, как устроен исходный текст, давший такое перекрытие скобок. Верхняя фигурная скобка, созданная командой `\overbrace`, ставится не над самой формулой $1 + 2 + 3$, а над ее фантомом. В результате команда `\overbrace` печатает фигурную скобку над пустым местом. Далее, вся эта конструкция стоит, в свою очередь, в аргументе команды `\lefteqn`, вследствие чего TeX считает, что места по горизонтали она не занимает. Поэтому формула $1 + \underbrace{2 + 3 + 4}$ начинается с того же места, что и фантом формулы $1 + 2 + 3$; в результате $1 + 2 + 3$ попадает аккурат под верхнюю скобку! Все это, конечно, — еще один пример «грязного трюка» (см. с. 72).

Если бы формула была не выключная, а внутритестовая, то этот трюк прошел бы не столь гладко. Дело в том, что команда `\lefteqn` всегда набирает формулы в `\displaystyle`, поэтому размер фантома, над которым ставилась скобка, мог в принципе не совпасть с размером реально печатаемого фрагмента формулы. Чтобы уж совсем себя обезопасить, следовало бы в этом случае аргумент команды `\lefteqn` начать с `\textstyle`.

С этой особенностью команды `\lefteqn` мы уже сталкивались: если вы присмотритесь к коммутативной диаграмме на с. 71, то заметите, что надписи при вертикальных стрелках вышли крупнее, чем при горизонтальных — как раз потому, что из-за команды `\lefteqn` они набирались шрифтом для выключных формул.

4.4. Снова об интервалах в формулах

Сейчас мы обсудим вкратце, какими правилами руководствуется TeX при расстановке интервалов в математических формулах. В стандартных ситуациях мы об этом не задумываемся, а полностью доверяем L^AT_EX'у. То, о чём мы будем говорить, пригодится, если мы пользуемся в формулах сложными конструкциями (например, конструируем знак двойного интеграла из двух знаков интеграла и «отрицательных пробелов», как на с. 77) и при этом не хотим подбирать верные интервалы экспериментально.

При наборе формулы TeX рассматривает ее как состоящую из частей одного из следующих типов:

Обыкновенный символ	например, <code>\alpha</code>
Бинарная операция	см. с. 44;
Бинарное отношение	см. с. 45;
Математический оператор	см. с. 46, 48;
Подформула	например, <code>{x^2}</code>
Знак препинания	, или ; или <code>\colon</code> или <code>\ldotp</code>
Скобка	

Здесь подформула — это любой фрагмент формулы, заключенный в фигурные скобки. Команда `\colon` задает двоеточие, рассматриваемое как знак препинания (двоеточие, набранное непосредственно, рассматривается $\text{\TeX}'$ ом как знак бинарного отношения), а команда `\ldotp` — точку, рассматриваемую как знак препинания (точка, набранная непосредственно, рассматривается как обычновенный символ). К бинарным отношениям (с точки зрения $\text{\TeX}'$ а) относятся также все стрелки (с. 45) и фрагменты формул, создаваемые командой `\stackrel`. При расстановке пробелов в формуле \TeX руководствуется тем, к какому из перечисленных типов относятся ее составные части: символы бинарных операций окружаются «средними пробелами» (теми, что вручную задаются командой `\:`), а символы бинарных отношений — «толстыми» пробелами (вручную, как мы помним, толстый пробел задается командой `\;`); впрочем, в стилях для индексов и индексов к индексам (см. предыдущий раздел) эти пробелы опускаются; после знака препинания в большинстве случаев ставится «тонкий» пробел, и т. д. Подформула (т. е. фрагмент формулы, заключенный в фигурные скобки) рассматривается $\text{\TeX}'$ ом почти так же, как обычный символ:

Сравните $2 + 3$ и $2+3$: во втором случае знак плюс является подформулой, а не символом бинарной операции.

Сравните $$2+3$$ и $$2\{+\}3$$: во втором случае знак плюс является подформулой, а не символом бинарной операции.

Кстати, с этим приемом (поставить фрагмент формулы в фигурные скобки, чтобы он рассматривался как обычный символ) мы уже сталкивались на с. 30, когда обсуждали, как задать в $\text{\TeX}'$ е десятичную дробь. Мы не будем вдаваться в точные правила расстановки пробелов (они перечислены в книге [3]). Для нас сейчас важнее то, что \TeX можно заставить рассматривать любой фрагмент формулы как бинарную операцию, бинарное отношение или математическую операцию: для этого надо применить команды `\mathbin`, `\mathrel` или `\mathop` соответственно. Вот примеры того, как работают эти команды.

Иногда возникает нужда в символе $\hat{\otimes}$, рассматриваемом как символ бинарной операции. Естественно, этот символ можно сгенерировать, написав `\hat{\otimes}`, но тогда вокруг этого символа будут неправильные пробелы:

Хотелось бы, чтобы в формуле $E\hat{\otimes}F$ были такие же пробелы, как и в $E \otimes F$.

Хотелось бы, чтобы в формуле $\$E\hat{\otimes}\{\\otimes\}F\$$ были такие же пробелы, как и в $\$E\\otimes F\$$.

Чтобы \TeX рассматривал $\hat{\otimes}$ не как обычный символ, а как символ бинарной операции, надо сделать так:

В формуле $E \hat{\otimes} F$ пробелы такие же, как и в $E \otimes F$.

В формуле
 $\$E\backslash mathbin{\backslash hat{\backslash otimes}}F\$$
 пробелы такие же, как и в $\$E\backslash otimes F\$$.

Если символ $\hat{\otimes}$ встречается в вашей рукописи часто, то вам вряд ли понравится всякий раз делать по 23 нажатия на клавиши для его набора. В этом случае очень удобно ввести для него собственное сокращенное обозначение (посмотрите начало гл. VII по поводу того, как это сделать).

Типичный пример использования команды $\backslash mathop$ — определение имени операции, записываемой прямым шрифтом (см. с. 47, где мы давали определение функции tg). Обозначения такого типа встречаются в математических текстах очень часто, и набора команд для них, предусмотренного $\text{\LaTeX}'ом$ (с. 46), вполне может не хватить; в этом случае, чтобы получить на печати, скажем, $\text{Ext}^1(E, F)$, надо написать:

$\$\\mathop{\\rm Ext}\\nolimits^1(E, F)\$$

Здесь $\backslash mathop$ необходимо для того, чтобы между именем «оператора» и тем, к чему он прилагается, автоматически вставлялся маленький дополнительный пробел, делающий формулу более читаемой:

Сравните $\sin x$ и $\sin x$.

Сравните $\$\\sin x\$$ и $\${\\rm sin}x\$$.

Что же касается \nolimits , то эта команда необходима для того, чтобы в выключных формулах (точнее, в «выключном стиле» — см. разд. 4.2) верхние и нижние индексы к «оператору» записывались именно как индексы, а не над и под ним, как «пределы» (см. с. 49).

- ε Если вы пользуетесь $\text{\LaTeX}'ом 2\varepsilon$, то в вашем распоряжении будет более удобное средство для определения новых операций, заданных прямым шрифтом. См. приложение E.

А вот пример, когда $\text{\TeX}'у$ надо объяснить, что некоторый сложный символ есть символ математического оператора. Предположим, нам понадобилась формула наподобие

$$\sum'_{x \in \Gamma} f(x).$$

Проблема тут в том, чтобы поставить штрих у знака суммы. Впрямую это сделать не удается:

$$\sum'_{x \in \Gamma} f(x).$$

\$\$ \\ \sum'_{x \in \Gamma} f(x). \\ \$\$

В самом деле, из сказанного на с. 50 вытекает, что наша запись равносильна такой:

\$\$ \\ \sum'_{x \in \Gamma} f(x). \\ \$\$

и в этой записи штрих рассматривается как предел суммирования. Не будем, однако, отчаиваться, а просто создадим новый оператор «сумма со штрихом»:

\$\$ \\ \mathop{\{}{\}}'_{{x \in \Gamma}} f(x). \\ \$\$

Можете проверить, что на сей раз все получается как надо. В этой записи очень существенно, что `\sum` взято в фигурные скобки: благодаря этому символ, генерируемый командой `\sum`, рассматривается `TeX`'ом просто как подформула, поэтому и штрих после него стоит где положено, а не там, где бывают пределы суммирования. Вся подформула `{\sum}'` передается в качестве аргумента команде `\mathop`, благодаря чему наш новый символ «сумма со штрихом» рассматривается как математический оператор и пределы суммирования (в выключной формуле) ставятся у него, где положено.

Здесь опять разумно создать сокращенное обозначение, которое заменяло бы эту громоздкую запись.

4.5. Дополнительные пробелы вокруг формул

Пришла пора и для последней из тех тонкостей набора формул, о которых упоминается в этой главе. Некоторые авторы и издатели считают, что математический текст выглядит понятнее, когда каждая формула окружена дополнительным пробелами справа и слева от нее. Для этих целей в `TeX`'е предусмотрен параметр `\mathsurround` (см. разд. I.2.6 по поводу `TeX`'овских параметров). Значение этого параметра — размер дополнительного пробела, вставляемого по обе стороны каждой внутритекстовой математической формулы (этот пробел не добавляется перед формулой, попавшей при печати в начало строки, и после формулы, попавшей в конец строки). При запуске `LATeX`'а значение этого

параметра равно нулю, так что расстояния между формулами и окружающим текстом такие же, как между словами в тексте. Можно, однако, присвоить параметру `\mathsurround` значение ненулевой длины. Например, если сказать в преамбуле

```
\mathsurround=2pt
```

то каждая формула будет окружена дополнительными пробелами по 2 пункта с обеих сторон.

Глава III

Набор текста

1. Специальные знаки

Большинство знаков препинания (точка, запятая, двоеточие и т. п.) набираются очевидным образом: точке в исходном тексте, например, соответствует типографская точка на печати. В этом разделе речь пойдет о знаках, требующих специального набора.

1.1. Дефисы, минусы и тире

При печати на пишущей машинке эти знаки по внешнему виду не отличаются. В издательских системах, основанных на \TeX 'е, различают дефис - (по-английски *hyphen*), короткое тире – (по-английски *en-dash*), длинное тире — (*em-dash*) и знак минуса – (обратите внимание, что он отличается от обоих тире).

Чтобы получить на печати дефис, короткое тире или длинное тире, надо в исходном тексте набрать один, два или три знака – соответственно. В русских текстах рекомендуется использовать длинное тире в качестве тире как такового⁵, а короткое тире — в сочетаниях типа «я вернусь через 2–3 часа» (в исходном тексте это выглядит как *через 2–3 часа*; обратите внимание на отсутствие пробелов вокруг тире). Длинное тире, напротив, должно быть окружено пробелами с обеих сторон (этого требует не \TeX , но принятые в России типографские правила).

Знак минуса, в отличие от короткого тире, встречается только в математических формулах, и там он, как вы помните, изображается просто знаком – (см. разд. I.3).

⁵Принятое в отечественной полиграфии тире несколько короче \TeX 'овского «длинного тире». Это нашло свое отражение в стилевом пакете *russianb* — см. приложение Ж. — Прим. ред.

У любознательного читателя может возникнуть вопрос, как так получается, что запись **—** в исходном тексте дает на печати всего-навсего две буквы «ж», а запись **--** дает тире, которое шире, чем два дефиса. Ответ: **TeX'**овские шрифты так устроены, что некоторые последовательности подряд идущих символов заменяются на печати на новый знак (говоря более техническим языком, в этих шрифтах допускаются лигатуры).

Наряду с тире, другой пример лигатур — это то, как выглядят в основных шрифтах сочетание букв **fi**: не так, как поставленные рядом **f** и **i**.

Близкое к этому явление — так называемый кернинг, когда некоторые пары букв, стоящие рядом, на печати автоматически сближаются: сравните **XO** (полученное на печати естественным образом) и **XO** (набранное со специальной командой, убирающей кернинг).

1.2. Кавычки

В отличие от пишущей машинки, книжный набор использует различные знаки для открывающей и закрывающей кавычек (вместо нейтрального знака ""). В англоязычных текстах открывающая кавычка изображается во входном тексте двумя подряд идущими обратными апострофами, закрывающая — двумя апострофами.

The “definitions” are trans- lations rather than explana- tions.	The ‘‘definitions’’ are translations rather than explanations.
--	---

Образование знака кавычек из двух апострофов — еще один пример лигатуры.

В русских текстах употребляются кавычки типа «елочки» и „лапки“. К сожалению, они отсутствуют в стандартном комплекте **TeX'**овских шрифтов; при установке системы **LATeX** вы должны выяснить, как именно они задаются в полученной вами русификации (и задаются ли вообще).

Если в тексте встречаются кавычки внутри кавычек, то, согласно типографским правилам, внутренние кавычки должны отличаться от внешних: в английских текстах снаружи ставятся двойные кавычки, задаваемые как ‘‘ и ’’, а внутри одинарные, задаваемые как ‘ и ’; в русских текстах можно, например, снаружи поставить «елочки», а внутри „лапки“. Если при этом наружная и внутренняя кавычка соседствуют, их надлежит разделить дополнительным небольшим пробелом. В **LATeX'**е этой цели служит команда **\,**. Пример (в русификации **LATeX'a**, использованной при написании этой книги, «елочки» задаются командами (с именами из русских букв!) **\лк** и **\pk**, а „лапки“ — командами **\glqq** и **\grqq**):

«„Карова“ или „корова“, как правильно?» — спросил он.

\лк\,\glqq Карова\grqq{} или
 \glqq корова\grqq{},
 как правильно?\лк\ ---
 спросил он.

Мы поставили «backslash с пробелом» после закрывающих кавычек, чтобы тире не напечаталось вплотную к кавычке (см. с. 18).

1.3. Многоточие

На пишущей машинке многоточие — это три точки подряд (каждая из которых имеет стандартную ширину буквы). При наборе это не так: для многоточия есть специальная команда `\ldots` или `\dots`.

Вместо «...» пишем: Нет, что-то
здесь не так ...

Вместо \лк ... \лк{} пишем:
Нет, что-то здесь не так\dots

1.4. Прочие значки

Знак параграфа набирается с помощью команды `\S`, знак © — с помощью команды `\copyright`; о том, что знаки \$ и & набираются с помощью команд `\$` и `\&`, мы уже говорили (см. разд. I.2.2); знак фунта стерлингов £ набирается с помощью команды `\pounds`. В разд. II.1.3 (а для пользователей L^AT_EX'а 2_E — и в приложении Д) перечислено еще несколько экзотических типографских значков, которые можно использовать в тексте. К сожалению, в стандартных L^AT_EX'овских шрифтах не предусмотрен знак №; желательно, чтобы он присутствовал в используемой вами русификации L^AT_EX'а. В русификации, использованной автором, этот знак задается командой `\номер` с именем из русских букв или командой `\№` с именем из латинских букв⁶. Кроме того, в тексте можно использовать и любой из великого множества математических символов, если оформить его как математическую формулу:

Я ♥ тебя.

Я \\$\heartsuit\\$ тебя.

Наконец, можно добраться до любого символа в любом из используемых L^AT_EX'ом текстовых шрифтов, если знать код этого символа. Для этих целей предназначена команда `\symbol`. Ее единственный обязательный аргумент — код символа. Для латинских букв и цифр эти коды совпадают с обычными ASCII-кодами:

Кошка по-английски бу-
дет cat.

Кошка по-английски будет
 \symbol{99}\symbol{97}\symbol{116}.

⁶См. также приложение Ж. — Прим. ред.

Коды остальных знаков в основных L^AT_EX'овских текстовых шрифтах приведены в приложении А.

Код символа можно указывать не только в десятичной системе, как в приведенном примере, но и в восьмеричной (тогда перед кодом надо поставить символ ') или шестнадцатеричной (перед кодом ставится символ ", «цифры» от А до Е должны быть прописными буквами). Например, записи `\symbol{122}`, `\symbol{'172}` и `\symbol{"7A}` на печати дадут одно и то же: букву z.

1.5. Подчеркивания, рамки

Чтобы подчеркнуть текст, используется команда `\underline`. У нее один обязательный аргумент — подчеркиваемый текст:

Это слово будет подчеркнуто.

Это слово будет
`\underline{подчеркнуто}`.

Подчеркнутый текст должен умещаться в одной строке.

Чтобы взять часть текста в рамку, используется команда `\fbox`:

Два слова будут в рамке.

Два слова будут `\fbox{в рамке.}`

Команда `\fbox` позволяет взять в рамку только фрагмент текста, умещающийся в одну строку. Чтобы взять в рамку фрагмент, состоящий из нескольких строк, надо воспользоваться командами, о которых пойдет речь в гл. VIII.

2. Промежутки между словами

2.1. Неразрывный пробел

Иногда необходимо обеспечить, чтобы два соседних слова не попали на разные строки. В этом случае между ними надо вставить «символ неразрывного пробела». Такая необходимость возникает, например, в сочетаниях типа «на с. 5»: нельзя отрывать номер страницы от сокращения «с.». Вот примеры:

Число овец в стаде обозначено буквой *x*. Да здравствует император Франц-Йосиф I! Муж и жена — одна сатана.

Число овец в стаде обозначено буквой "*x\$*". Да здравствует император Франц-Йосиф I!
Муж и жена --- одна сатана.

В последнем из этих примеров мы поставили неразрывный пробел, поскольку согласно отечественным полиграфическим правилам строка не должна начинаться с тире.

2.2. Промежутки между предложениями

В обычном режиме \TeX выравнивает справа строки абзаца, при необходимости делая переносы и слегка растягивая или сжимая промежутки между словами. Промежутки между предложениями при этом сами по себе шире и являются более растяжимыми, чем между словами внутри предложения. Посмотрите внимательно на следующий пример (из «Винни-Пуха»):

North Pole. Discovered by Pooh. Pooh found it.

Такая печать соответствует английским типографским правилам. В русских текстах, однако, промежутки между словами и между предложениями отличаться не должны. Чтобы так и было, следует включить в преамбулу команду `\frenchspacing`.

Если среди русского текста встречается фрагмент, написанный по-английски, то можно командой `\nonfrenchspacing` восстановить действие английского правила относительно межсловных промежутков. Когда английский текст кончится, надо восстановить действие российского правила командой `\frenchspacing` (другой вариант: заключить английский фрагмент вместе с командой `\nonfrenchspacing` в группу — по выходе из группы действие команды `\nonfrenchspacing` забудется).

Для читателей, которым необходимо набирать английские тексты, объясним более подробно правила расстановки промежутков в тех случаях, когда команды `\frenchspacing` не было.

Чтобы отличить промежутки между словами от промежутков между предложениями, \TeX не проводит синтаксического разбора, но применяет следующие правила:

1) Пробел увеличивается после:

- точки, вопросительного знака, восклицательного знака (в максимальной степени);
- двоеточия (несколько меньше);
- точки с запятой (еще меньше);
- запятой (совсем чуть-чуть).

- 2) Если последняя из букв, встретившихся перед одним из упомянутых в пункте 1 знаков препинания, была прописной, то пробел после этого знака препинания не увеличивается.
- 3) Если после одного из упомянутых в пункте 1 знаков препинания следует закрывающая скобка (круглая или квадратная) или закрывающие кавычки, а затем — пробел, то этот пробел увеличивается.

Смысл правила 2 в том, что точка после прописной буквы чаще всего обозначает не конец предложения, а конец чьих-то инициалов.

Как это и бывает обычно с «машинными эвристиками», сформулированные правила иногда приводят к неверным результатам: точка после строчной буквы может встретиться и в середине предложения, например, в сокращении, а точка после прописной буквы может, напротив, попасть в конец предложения. В этих случаях надо следующим образом помочь TeX'у сделать правильные пробелы:

- Если точка после строчной буквы не заканчивает предложения, то после нее следует поставить команду `\`` (backslash с пробелом), генерирующую обычный пробел между словами (см. с. 18).
- Если точка (или любой другой из перечисленных в пункте 1 знаков препинания) после прописной буквы заканчивает предложение, то перед ней следует поставить команду `\@` — тогда пробел после точки будет обычным образом увеличен.

Вот примеры:

The case when the index is 0, i.e. C
is linearly normal, is quite simple.
This research was supported by
NSF. The author is grateful to
Prof. Smith.

The case when the index is "\$0\$,
i.e. \\$C\\$ is linearly normal,
is quite simple.

This research was supported by
NSF\@. The author is grateful
to Prof.^Smith.

Наконец, последнее правило относительно увеличения пробелов: если пробел задан как неразрывный с помощью символа `"`, то он не увеличивается, невзирая ни на какие предшествующие знаки препинания.

2.3. Установка промежутков вручную

Как команда «backslash с пробелом» `\``, так и символ неразрывного пробела `"` генерируют пробел, но делать пробелы вручную с помощью набора чего-нибудь вроде `""` или `\ \ \`` неразумно, поскольку эти пробелы, как правило, могут растягиваться или сжиматься ради выравнивания строк, и вы не сможете проконтролировать реальный размер пустого пространства, полученного таким способом.

Чаще всего требуется получить промежуток величиной в один или два em (см. с. 25). Для этого служат команды `\quad`, дающая промежуток в 1 em, и `\quad\quad`, дающая промежуток в 2 em. Команда `\enskip` дает промежуток, в два раза меньший, чем `\quad` (в стандартных шрифтах он равен ширине цифры). Про команду `\,` («backslash с запятой») уже шла речь в разд. 1.2.

Если необходимо задать промежуток с указанием конкретной длины, можно воспользоваться командой

`\hspace{длина}`

Если этот промежуток должен сохраняться также и в начале (или конце) строки, используется команда `\hspace*` вместо `\hspace`. Указание длины состоит из числа и названия единицы (см. с. 25), например, `\hspace{1.5cm}`.

Пользуясь командами типа `\quad`, не сделайте лишнего пробела (разумеется, кроме того, который ограничивает имя команды, состоящей из букв). Вот примеры того, как надо и как не надо делать:

Здесь 1em промежутка.

Здесь 1em промежутка.

Здесь больше, чем надо.

Здесь \quad 1em промежутка. \\

Здесь \quad{}1em промежутка. \\

Здесь \quad{} больше,

чем надо. \\

В этом примере используется еще команда `\\"`, начинающая новую строку (см. подробнее на с. 109).

3. Диакритические знаки и прочее

Во многих языках используются буквы с дополнительными знаками, размещающимися над или под буквой (они называются диакритическими знаками). Кроме того, в ряде языков, использующих латинский алфавит, есть специальные дополнительные буквы. В TeX'е имеются команды для набора таких букв с диакритическими знаками из почти всех европейских языков. Команды для получения диакритических знаков собраны в нижеследующей таблице, где знаки проставлены, для примера, при букве «е».

Набрано	Вышло	Набрано	Вышло
\`e	è	\`e	é
\^e	ê	\^e	ë
\=e	ē	\=e	ê
\u{e}	ě	\v{e}	ě
\H{e}	�	\H{e}	�
\c{e}	�	\c{e}	�
\b{e}	�	\t oo	��

В L^AT_EX'е 2_E определена еще команда `\r`:

Набрано	Вышло
\r{e}	�

В следующей таблице вы найдете команды для набора букв специального вида, а также перевернутых вопросительного и восклицательного знаков (последние используются в испанском языке).

Набрано	Вышло	Набрано	Вышло
\oe	œ	\OE	Œ
\ae	æ	\AE	Æ
\aa	å	\AA	Å
\o	ø	\O	Ø
\l	ł	\L	L
\i	ı	\j	j
\ss	ß		
!‘	¡	?‘	¿

Обратите внимание на команды \i и \j в этой таблице: они нужны для того, чтобы ставить диакритические знаки над буквами i и j. Если просто сказать, допустим, \=i, то получится ī, а это не то, что требуется. Правильно писать \=\i. Вот несколько примеров.

La leçon était très difficile.

Ель на ёжика похожа: ёж в иголках, ёлка тоже.

¡No pasarán!

La le\c{c}on \’etait
tr\`es difficile.

Ель на \"ежика похожа: \"ех
в иголках, \"елка тоже.
! 'No pasar\'an!

Для набора обычных русских текстов из всего этого великолепия нужны фактически только две команды: \' (чтобы ставить ударения) и \" (для буквы ё, если в вашей русификации L^AT_EX'а для нее не отведено особого места).

Команды \c, \' и т. д. имеют один обязательный аргумент — букву, над (или под) которой надо ставить диакритический знак. Внимательный читатель может заметить тут противоречие со сказанным на с. 23: ведь обязательный аргумент должен быть заключен в фигурные скобки, а в нашей таблице в записях вроде \'e или \"e фигурных скобок нет. На самом деле противоречия нет, просто на с. 23 было сказано не все: если у L^AT_EX'овской команды предусмотрен один обязательный аргумент и в исходном тексте после имени команды непосредственно следует буква, то в качестве аргумента будет воспринята именно эта буква (ср. с. 57). Так что можно было бы, в принципе, не ставить букву в фигурные скобки и при командах наподобие \c или \u (но психологически приятнее, когда слово, которое на печати выйдет без пробелов, не будет содержать пробелов и в исходном тексте):

façade — он и есть façade.

fa\c{c}ade ---
он и есть
fa\c cade.

Команды из второй таблицы аргументов не требуют; что же касается значков ‘! и ?’, то это вообще не команды, а лигатуры (см. текст мелким шрифтом на с. 87).

Над буквами в математических формулах также приходится ставить надстрочные знаки, но описанные в настоящем разделе команды для этого непригодны; команды, делающие это в формулах, описаны в разд. II.2.8.

В заключение нашего обсуждения диакритических знаков отметим вот что. Хотя описанные в этом разделе команды дают возможность набирать все знаки алфавита почти любого европейского языка, это еще не значит, что вы так просто сможете набирать на TeX’е длинные тексты на этих языках. Дело в том, что при отсутствии ориентированной на соответствующий язык таблицы переносов TeX будет, скорее всего, переносить слова в этих текстах неправильно. Таблица переносов для русского языка в русификации LATEX’а имеется, а для английского языка такая таблица присутствует в TeX’е изначально. Если же вам придется набирать хотя бы абзац текста на каком-то другом языке, а соответствующей таблицы переносов у вас нет, то лучше переключиться на режим, в котором переносы вообще запрещены. Как это сделать, будет рассказано в разделе, посвященном абзацам.

4. Смена шрифтов в тексте: LATEX 2.09

В первой главе мы уже сталкивались с командами, изменяющими текущий шрифт. Давайте рассмотрим их систематически.

Начнем мы с такого предупреждения, которое не требуется профессиональным полиграфистам, но не повредит остальным читателям:

Не увлекайтесь переключением шрифтов! Чем меньше различных видов шрифта использовано в тексте, тем легче его читать и тем красивее он выглядит.

Обычно шрифт, отличный от используемого в основной части текста, применяется для выделения каких-то частей этого текста. Например, шрифтом выделяют заголовки разделов; по этому поводу вам беспокоиться незачем, поскольку для таких выделений LATEX выбирает шрифт автоматически (если, разумеется, вы оформляете разделы текста с помощью команд, описываемых в следующей главе, а не пытаетесь сделать это вручную). Кроме того, может потребоваться выделить шрифтом не заголовок, который LATEX делает сам, а какую-то выбранную вами часть текста. На этот случай в LATEX’е предусмотрена команда \em. Если текущий шрифт прямой, то эта команда переключает шрифт на

курсивный, а если имеет наклон, то на обычный прямой светлый. Поскольку в некоторых ситуациях *LATEX* выбирает шрифт за вас (в заголовках, колонитулах, в текстах «теорем» и т. п.), рекомендуется для выделения текста использовать в первую очередь именно эту команду.

Употребляя команду `\textit{}` (а также команды, устанавливающие «наклонный» шрифт), необходимо учитывать, что при соседстве слова, набранного шрифтом с наклоном (курсивом в частности) и слова, набранного прямым шрифтом, последняя буква «наклонного» и первая буква «прямого» слова могут слишком сблизиться, что на печати выглядит некрасиво. Взгляните еще раз на с. 21, где последняя буква слова «восстановится» сближается с первой буквой слова «шрифт». Чтобы избежать этого явления, необходимо после последней буквы слова, которое будет набрано наклонным шрифтом, поставить команду `\!/`; она создаст после буквы небольшой дополнительный пробел (зависящий от шрифта и от буквы), который скомпенсирует наклон и предотвратит нежелательное сближение со следующей буквой. Если фрагмент текста, имеющий наклон, завершается точкой или запятой, то после них ставить `\!/` не нужно: требуемый эффект достигается за счет места, занимаемого в тексте этим знаком. Проиллюстрируем все сказанное примером.

Этот текст выделен; внутри выделенного текста можно тоже делать выделения тем же способом; теперь снова обычный текст. Если выделенный текст кончается запятой, то предосторожности не нужны.

Этот текст `\textit{}` выделен; внутри выделенного текста можно тоже делать `\textit{выделения\!}` тем же способом; теперь снова`\!` обычный текст. Если выделенный текст кончается `\textit{запятой,}` то предосторожности не нужны.

Если команда `\!/` окажется после текста, который в результате действия `\textit{}` оказался набран прямым шрифтом, то никакого вредного действия это не окажет.

Если команда `\!/` поставлена между двумя буквами, дающими на печати лигатуру (см. с. 87), то вместо лигатуры на печати получатся две эти буквы по отдельности; если эту команду поставить в слове между двумя буквами, между которыми в текущем шрифте предусмотрен кернинг, то кернинг между этими буквами будет отключен.

Теперь перечислим все существующие в стандартном комплекте *LATEX*'а начертания⁷ и размеры шрифта. Начнем с начертаний. Они

⁷Не путать с термином «начертание», используемым в классификации шрифтов *LATEX*'а 2ε; см. с. 98. — Прим. ред.

Таблица III.1

Команда	Название начертания
\bf	полужирный шрифт (boldface)
\it	курсив (<i>italic</i>)
\sl	наклонный шрифт (<i>slanted</i>)
\sf	рубленый шрифт (<i>sans serif</i>)
\sc	КАПИТЕЛЬ (SMALL CAPS)
\tt	имитация пишущей машинки (<i>typewriter</i>)
\rm	прямой светлый шрифт (<i>roman</i>)

перечислены в табл. III.1 вместе с командами, задающими переключение на них. Шрифты начертания \tt обладают специфическими свойствами: все символы в них имеют одинаковую ширину, а промежутки между словами жестко фиксированы и не обладают растяжимостью. Применяются эти шрифты для изображения компьютерных программ, фрагментов TeX'овских исходных текстов и т. п.

А теперь перечислим возможные размеры шрифтов: Примеры использования команд переключения шрифтов вы видели на с. 21.

Команды, меняющие размер, одновременно устанавливают начертание *roman* (прямой светлый шрифт). Поэтому полужирный большой шрифт требует не команд \bf\large, а команд \large\bf.

- ε В LATEX'e 2ε команды для установки размера и начертания можно давать в любой последовательности. Кроме того, в LATEX'e 2ε доступно больше, чем в LATEX'e 2.09, различных начертаний шрифтов, а команды переключения шрифтов организованы по-иному.
- ε См. по этому поводу следующий раздел.

Скорее всего, вы будете давать команды, переключающие шрифт, внутри группы. При этом не сделайте лишнего пробела до или после фигурной скобки, как в следующем примере:

Вот так получится плохо.

Вот так будет правильно.

Вот {\bf так } получится плохо. \\

Вот {\bf так} будет правильно.

Реальный размер шрифтов, задаваемых командами \large, \small и т. п. зависит от стиля. В стандартных стилях с основным шрифтом кегля 12 команды \huge и \Huge задают один и тот же размер (кегль 25).

С командами, меняющими размер шрифта, связана одна распространенная ошибка. Если вы набрали шрифтом измененного размера (скажем, \small или \footnotesize) целый абзац, то в момент, когда TeX

Таблица III.2

Команда	Название размера
\tiny	Малюсенький
\scriptsize	Очень маленький (как индексы)
\footnotesize	Маленький (как сноски)
\small	Мелкий
\normalsize	Нормальный
\large	Большой
\Large	Очень большой
\LARGE	Совсем большой
\huge	Громадный
\Huge	Грандиозный

видит пустую строку (или команду `\rag` — см. с. 117), этот шрифт не должен быть еще переключен на обычный, иначе интервалы между строками получатся неправильные. Вот пример того, как надо и как не надо действовать:

Мы закрываем группу и возвращаемся к обычному шрифту только после пустой строки, завершающей абзац.
Вот шрифт обычного размера.

{\footnotesize Мы закрываем группу и возвращаемся к обычному шрифту только после пустой строки, завершающей абзац.

Вот шрифт обычного размера.
}

Здесь мы вернулись к обычному шрифту раньше времени. Результат — межстрочные интервалы слишком велики.

Вот шрифт обычного размера.

{\footnotesize Здесь мы вернулись к обычному шрифту раньше времени. Результат — межстрочные интервалы слишком велики.}

Вот шрифт обычного размера.

Для любознательных объясним, откуда берется этот эффект. Команды, меняющие размер шрифта, изменяют не только текущий шрифт, но и расстояние

между строками (увеличивают, если размер шрифта увеличивается, и уменьшают, если размер шрифта уменьшается). Однако же определение расстояний между строками происходит только на заключительном этапе верстки абзаца. Поэтому, если \TeX набирал абзац во время действия, скажем, команды \small , а команду «сверстать абзац» (например, пустую строку) увидел тогда, когда действие команды \small уже кончилось, то между строками, набранными мелким шрифтом, будут установлены те же расстояния, как между строками, набранными шрифтом обычного размера.

5. Смена шрифтов в тексте: $\text{\LaTeX}\ 2\epsilon$

Все команды смены шрифтов, описанные в предыдущем разделе, поддерживаются и в $\text{\LaTeX}'e\ 2\epsilon$ и работают в нем точно так же (за тем приятным исключением, что команды \small , \large и т. п. действительно меняют именно размер, а начертание всегда сохраняют). Наряду с этим, однако же, $\text{\LaTeX}\ 2\epsilon$ предоставляет много новых шрифтовых возможностей. Сначала мы опишем, какие новые шрифты есть в стандартной поставке $\text{\LaTeX}'a\ 2\epsilon$, а затем — какими командами они управляются.

Каждый из доступных в $\text{\LaTeX}'e\ 2\epsilon$ текстовых шрифтов полностью характеризуется следующими четырьмя атрибутами: семейством (family), насыщенностью (series), начертанием (shape) и размером (size). Что такое размер, читатель разберется самостоятельно, а смысл остальных атрибутов таков:

семейство означает примерно (но не в точности) то же, что отечественный термин « гарнитура »; в стандартной поставке определены семейства `rmfamily` (шрифты с засечками), `sffamily` (шрифты без засечек) и `ttfamily` (шрифты типа « пишущая машинка »);

насыщенность определяет ширину и жирность шрифта. В стандартной поставке возможны насыщенности средняя (`mdseries`) и полуожирная (`bfseries`);

начертание бывает прямое (`upshape`), курсивное (`itshape`), наклонное (`slshape`) и капиталь (`scshape`).

Семейство, насыщенность, начертание и размер шрифта могут различным образом сочетаться. Вот, например, предложение, набранное шрифтом семейства «без засечек» (`sffamily`), полужирной насыщенности (`bfseries`), прямого начертания (`upshape`) и размера `large`.

Каждый из шрифтовых атрибутов можно менять независимо от остальных. Разберем, какие команды для этого предусмотрены.

Таблица III.3

Без аргументов:	С аргументом:	На печати выйдет:
{\rmfamily Шрифт}	\textrm{Шрифт}	Шрифт
{\sffamily Шрифт}	\textsf{Шрифт}	Шрифт
{\ttfamily Шрифт}	\texttt{Шрифт}	Шрифт
{\mdseries Шрифт}	\textmd{Шрифт}	Шрифт
{\bfseries Шрифт}	\textbf{Шрифт}	Шрифт
{\upshape Шрифт}	\textup{Шрифт}	Шрифт
{\itshape Шрифт}	\textit{Шрифт}	Шрифт
{\slshape Шрифт}	\textsl{Шрифт}	Шрифт
{\scshape Шрифт}	\textsc{Шрифт}	ШРИФТ

Все команды для изменения размера вам уже известны: это десять команд, от `\tiny` до `\Huge`, перечисленных в предыдущем разделе. Каждая из команд для изменения остальных атрибутов существует в L^AT_EX'е 2_ε в двух вариантах:

- 1) в виде команды без аргументов, меняющей атрибут текущего шрифта вплоть до того момента, пока он не будет изменен другой командой (или пока не закончится группа, если атрибут менялся внутри группы);
- 2) в виде команды с одним аргументом (помещаемым, как водится, в фигурные скобки), меняющей атрибут шрифта только у своего аргумента (т. е. у текста в фигурных скобках).

Имя команды без аргумента совпадает с английским названием соответствующего атрибута (например, `\sffamily` или `\scshape`); имя команды с аргументом состоит из слова `text`, к которому добавлены две буквы, описывающие атрибут (например, `\textsf` или `\textsc`).

В следующей ниже табл. III.3 перечислены все команды для смены атрибутов в обоих вариантах. Обратите внимание, что в трех строках этой таблицы шрифт совпадает с основным шрифтом текста: это строки, в которых стоят команды, устанавливающие семейство «с заческами», среднюю насыщенность или прямое начертание. Поскольку текущий шрифт и так обладает этими атрибутами, то от соответствующей команды он не меняется.

Вот пример применения этих команд с включением шрифтов, недоступных L^AT_EX'у 2.09:

Сменим сначала начертание, затем семейство, затем размер, затем насыщенность, затем все вернем на место.

Сменим сначала `\slshape` начертание, затем `\ttfamily` семейство, затем `\small` размер, затем `\bfseries` насыщенность, затем `\upshape\mdseries` `\rmfamily\normalsize` все вернем на место.

Некоторым сочетаниям атрибутов никакого реального шрифта не соответствует. В этом случае затребованный, но отсутствующий шрифт заменяется на другой (по возможности, с близкими атрибутами). В нашем примере, в частности, не существует шрифта с атрибутами `\ttfamily` и `\bfseries`, поэтому \LaTeX действует так, словно была дана команда `\mdseries`. О каждой такой замене $\text{\LaTeX}\ 2\varepsilon$ выдает сообщение в процессе трансляции.

А вот пример, когда для смены атрибутов шрифта используются команды с аргументом:

Перейдем на **жирный** шрифт в **курсивном** начертании (временно, конечно же).

Перейдем на `\textbf{жирный}` шрифт в `\textit{курсивном}` начертании} (временно, конечно же).

Команда `\em`, задающая выделение текста (курсивом на фоне прямого шрифта и прямым шрифтом на фоне наклонного) также имеет в $\text{\LaTeX}'e\ 2\varepsilon$ аналог в виде команды с аргументом: команда `\emph` имеет обязательный аргумент — выделяемый текст. Пример:

Можно выделить несколько слов в тексте. В этом тексте тоже можно **кое-что** выделить.

Можно `\emph{выделить}` несколько слов в тексте. `\bfseries` В этом тексте тоже можно `\emph{кое-что}` выделить.

Обратите внимание, что на фоне полужирного шрифта (`\bfseries`) команда `\emph` поменяла только атрибут «начертание», сменив его на курсивное; в $\text{\LaTeX}'e\ 2.09$ команда `\em` сменила бы шрифт на светлый курсив.

Кроме разного принципа действия, у команд `\emph` и `\em` есть еще одно важное отличие: `\emph` автоматически вставляет «коррекцию наклона» после выделенного текста, так что при пользовании ею отпадает необходимость вручную добавлять `\V`. Кстати, команды `\textit` и `\textsl` также автоматически добавляют коррекцию наклона к своему аргументу.

После многочисленных изменений атрибутов шрифта хочется вернуться к обычному шрифту «одним махом», не устанавливая заново все четыре атрибута. Для этих целей предусмотрена команда `\normalfont`, переключающая шрифт на «нормальный» — основной шрифт документа. Наряду с ней есть, как водится, и команда с одним аргументом `\textnormal`, печатающая текст, являющийся ее аргументом, основным шрифтом.

6. Сноски

Чтобы сделать сноска к какому-то месту в тексте, достаточно использовать команду `\footnote` с одним обязательным аргументом — текстом сноски. В стандартных стилях `LATEX'a` сноски⁸ нумеруются подряд на протяжении всей главы или даже (в стиле `article`) всего документа. В исходном тексте предыдущий фрагмент выглядел так:

```
сноски\footnote{Вроде этой} нумеруются ...
```

В разделе, посвященном «счетчикам», мы расскажем о том, какие возможности есть для того, чтобы помечать сноски по-другому.

Если после слова, к которому делается сноска, должен стоять знак препинания, то в исходном тексте его надо поставить *после* закрывающей фигурной скобки, ограничивающей аргумент команды `\footnote`.

Текст сноски может состоять из нескольких абзацев; в этом случае они, как обычно, разделяются пустой строкой.

К сожалению, весьма непросто заставить `TEX` автоматически нумеровать сноски так, чтобы нумерация начиналась заново на каждой странице. В `LATEX'e`, в частности, такая возможность не предусмотрена. Если вы готовы пожертвовать автоматической нумерацией сносок, то можно воспользоваться командой `\footnote` с необязательным аргументом. Этот необязательный аргумент ставится (в квадратных скобках) перед обязательным³³. Предыдущий фрагмент выглядел в исходном тексте так:

```
обязательным\footnote[33]{Видите, какой ... }
```

При использовании команды `\footnote` с необязательным аргументом автоматическая нумерация сносок «не сбивается»: предыдущая сноска имела номер 8, затем мы искусственно создали сноsku номер 33, а следующая сноска⁹ будет иметь номер 9.

⁸Вроде этой

³³Видите, какой интересный номер получился!

⁹Вот эта.

В случае, если вы хотите сделать сноска к тексту, входящему в «блок» (например, в аргумент команды `\mbox`; в гл. VIII мы расскажем о том, что такое блок в общем случае и какие команды генерируют блоки), команда `\footnote` непригодна. Вот как надо ставить сноски в этом случае:

```
Роман \mbox{\лк Три\footnotemark{}}
мушкетера\пк}\footnotetext{А не четыре!} написал Дюма.
```

В этом случае получится нормальная сноска, информирующая нас, что Роман «Три¹⁰ мушкетера» написал Дюма. Если бы мы просто написали `\footnote`, то увидели бы на печати только номер, но не саму сноsku.

Если вы к тому же хотите вручную задать номер сноски к тексту, входящему в «блок», то нужно задать этот номер дважды: первый раз в качестве необязательного аргумента команды `\footnotemark`, а второй раз — в качестве необязательного аргумента команды `\footnotetext` (необязательный аргумент этой команды должен идти *перед* обязательным):

```
Роман \mbox{\лк Три\footnotemark[99]
мушкетера\пк}\footnotetext[99]{А не четыре!}
написал Дюма.
```

7. Абзацы

Чтобы \TeX сверстал абзац, никаких специальных усилий прилагать не нужно: достаточно оставить в исходном тексте пустую строку, указывающую $\text{\TeX}'$ у на конец абзаца. В этом разделе речь пойдет о тех «нештатных ситуациях», которые иногда при этом возникают.

Обычно абзацы делаются выровненными по правому краю; при необходимости промежутки между словами растягиваются или сжимаются, а в словах делаются переносы. \TeX выбирает из всех вариантов разбиения текста абзаца на строки оптимальный; при этом и для сжатия, и для растяжения промежутков между словами есть пределы, которые \TeX старается не превышать.

Если это удается, то получается аккуратный абзац. Нас же сейчас больше волнует, что происходит в том случае, когда это не удается.

Прежде всего заметим, что «предел сжимаемости» строки не превышается $\text{\TeX}'$ ом ни при каких условиях¹¹; что бы ни было, строки не станут более тесными, чем им позволяют параметры шрифта. Поэтому

¹⁰А не четыре!

¹¹Если не предпринимать для этого специальных усилий.

строки, которые не удалось ужать, остаются чересчур длинными (при этом, естественно, они выходят на поля документа). С другой стороны, предел растяжимости, за неимением ничего лучшего, может быть превышен. При этом получается то, что полиграфисты называют жидкими, или разреженными строками:

Весьма и весьма жидкая строка.

О каждой из этих двух неприятностей ТЕХ сообщает в процессе трансляции (эти сообщения появляются на экране, и, кроме того, записываются в log-файл — специальный файл с тем же именем, что у файла, который обрабатывался системой, и расширением log).

Предположим, вы получили строку, выходящую на поля (в нашем примере она отмечена черным прямоугольником):

Еще одно правило относительно увеличения пробелов: если

пробел задан как неразрывный, то он не увеличивается, не взирая ни на какие предшествующие знаки препинания.

При получении строк, выходящих на поля, выдается на экран и записывается в файл с расширением log сообщение наподобие такого:

```
Overfull \hbox (1.77339pt too wide)
in paragraph at lines 7--12
\elvrm ... уве-ли-че-ния про-бе-лов: если
```

```
\hbox(7.54332+2.12917)x310.72488, glue set - 1.0
.\hbox(0.0+0.0)x17.0
.\elvrm E
.\elvrm щ
.\elvrm е
.\glue 3.65 plus 1.825 minus 1.21666
.etc.
```

Для нас с вами существенна информация, содержащаяся в первых трех строках этого сообщения (дальнейший загадочный текст, записываемый только в log-файл, но не на экран, описывает, как именно «сделана» неудачная строка; такие вещи могут представлять интерес только для профессиональных ТЕХников). Давайте разберем, что в них написано. Сначала идет надпись «Overfull \hbox», указывающая, что произошло «переполнение» (overfull) строки. В скобках указано, на какое именно расстояние строка выходит за край: на 1.77339 пунктов. (Напомним, что пункт примерно равен одной трети миллиметра — см. с. 25.) Далее сказано, что переполнение произошло при верстке абзаца (слова «in

`paragraph»), а затем указаны номера строк исходного файла, в которых был записан этот абзац.`

Наконец, в третьей строке этого сообщения приведен фрагмент неудачной строки вблизи ее конца (конца не в исходном тексте, а на печати). Обратите внимание, что в некоторые слова вставлены дефисы: они показывают те места, в которых \TeX в принципе мог бы сделать переносы. Когда мы взглянемся внимательнее, то станет ясна и причина катастрофы: в слове `если`, которым заканчивается строка, дефиса не стоит вообще; значит, \TeX не смог найти подходящего места для переноса и оказался перед неприятным выбором: либо перенести это «если» целиком на другую строку (что, видимо, вызвало бы проблемы в других местах), либо оставить его на этой строке и создать `overfull`. Выбрано было второе (ниже мы объясним, как можно в какой-то мере управлять этим выбором).

- ε $\text{\LaTeX} 2\epsilon$, щадя пользователей, не выдает непонятных простому смертному подробностей об устройстве переполненной строки даже в `log`-файл.

Сообщения о жидких строках выглядят так:

```
Underfull \hbox (badness 2318)
in paragraph at lines 940--942
\elvrm мен-ты спис-ка ну-ме-ру-ют-ся
```

(в `log`-файле эти сообщения имеют и продолжение, аналогичное тому, что мы видели в сообщении о переполнении; мы не стали воспроизвести его еще раз). Главных элементов в этих сообщениях три:

- само слово `Underfull`, указывающее, что речь идет о жидкой строке;
- указание на строки исходного текста, в которых находится абзац с жидкой строкой (в нашем случае 940–942);
- численная характеристика того, насколько разрежена строка (по-английски это число называется `badness`). В нашем случае это число равно 2318; вскоре мы обсудим, что оно значит.

Итак, мы выяснили, какие могут быть неприятности при верстке абзацев и как \TeX сообщает нам об этих неприятностях. Вся оставшаяся часть этого раздела посвящена тому, как с этими неприятностями бороться.

7.1. Переносы

Вероятно, читатель был шокирован тем, что в нашем примере с overfull'ом TeX не смог найти, где сделать перенос в слове «если». Дело, в частности, в том, что обычно TeX не делает переносов, при которых от слова отрываются две последние буквы. Это соответствует нормам английской орфографии, но для русского языка такое ограничение ненужно. Поэтому при работе с русскими текстами можно (и нужно) установить такой режим, в котором перенос двух последних букв допустим. Устанавливается этот режим с помощью изменения параметра \righthyphenmin (см. с. 22 по поводу TeX'овских параметров). Значение этого параметра — целое число, равное наименьшему количеству букв в слове, которые можно переносить на следующую строку. Стало быть, если написать в файле

```
\righthyphenmin=2
```

то при обработке всех абзацев, кончающихся после этой команды, переносы с отрывом двух последних букв будут разрешены. Если вам встретится абзац английского текста, то перед завершающей его пустой строкой дайте команду

```
\righthyphenmin=3
```

дабы не сделать неверных переносов в английских словах.

Общее облегчение режима переносов с помощью \righthyphenmin может, тем не менее, не помочь преодолеть встретившийся вам overfull: не исключено, что TeX действительно не знает, как перенести какое-то слово. Таких слов не очень много, но они встречаются. Кроме того, TeX не делает автоматических переносов в словах, содержащих диакритические знаки (для русского языка — ударения над буквами или букву ё), а также в словах, в которых присутствуют наряду с буквами цифры, знаки препинания и т. п. Далее, если в слове присутствует дефис, то TeX сможет сделать в нем перенос только на месте этого дефиса (слово «генерал-губернатор» будет перенесено так, что генерал- останется в конце строки, а губернатор начнет следующую строку). Что делать в таких случаях, когда автоматически вставляемых TeX'ом переносов не хватает?

Совет номер один — попробуйте немного отредактировать абзац, изложив в нем мысли по-другому. Обычно после небольших изменений в абзаце неприятности с переносами исчезают; как показывает опыт, качество текста при этом тоже зачастую улучшается (с точки зрения языка, а не TeX'a). Пусть, однако, улучшать изложение дальше некуда,

а абзац все равно получается неудачный. Посмотрим, что еще можно сделать, чтобы избавиться от переполнения.

Если TeX не может перенести слово, перенос которого по правилам русского языка возможен, то есть два способа указать TeX'у на это обстоятельство.

Во-первых, существует «одноразовый» способ указать TeX'у места переносов в слове. Это делается с помощью команды \- таким, например, образом:

`на\-\\"ем\-\ник`

Команда \- означает, что данное слово можно переносить в тех и только тех местах, где стоят знаки \- (хотя бы и вопреки тому, что диктует TeX'овский алгоритм переноса). Она годится для любых слов (с буквой ё, цифрами и т. д.). Однако при этом TeX не запоминает, какие слова и в каких местах позволила ему перенести команда \-. Если, например, то же слово «найдник» встретится в тексте еще раз, TeX по-прежнему будет считать, что переносить его нельзя.

Во-вторых, если слово, которое надо переносить не в тех местах, которые находит алгоритм переноса, встречается в тексте неоднократно, имеет смысл указать это TeX'у «раз и навсегда» (в пределах данного документа). Для этого предназначена команда \hyphenation. В качестве ее аргумента указываются слово или слова, в которых дефисами обозначены разрешенные места переносов. Например:

`\hyphenation{вклю-чен об-ласть}`

Теперь слова «включен» и «область» всегда будут переноситься так, как было указано (хотя бы и вопреки тому, что диктует алгоритм переноса). Если в слове, указанном в качестве аргумента команды \hyphenation, дефисов не поставить, то это будет означать, что переносить его вообще нельзя. Разумное место для команды \hyphenation — преамбула документа.

Слова, указанные в аргументе команды \hyphenation, должны быть разделены пробелами. Не забывайте, что конец строки — тоже пробел, так что слова можно располагать и в нескольких строках. Пустой строки в аргументе \hyphenation быть не должно. В качестве аргумента команды \hyphenation нельзя указывать слова с диакритическими знаками или небуквенными символами.

Слова в исходном тексте, в которые вставлены \-, будут переноситься именно там, где указано этими командами, невзирая на то, что говорит команда \hyphenation.

Бывают случаи, когда избавиться от переполнения не помогают даже идеально расставленные переносы: например, если в конец строки

попадает слово, которое переносить нельзя, вроде «гвоздь», или не допускающая переноса математическая формула (кстати, если в неудачной строке присутствует нечто более сложное, чем просто текст, \TeX в сообщении о переполнении заменяет это на пару квадратных скобок \square). Что делать в таких случаях, рассказано в последующих разделах.

7.2. Команда $\backslash sloppy$ и параметр $\backslash emergencystretch$

Существует простой и грубый способ раз и навсегда избавиться от переполнений. Для этого достаточно включить в файл команду $\backslash sloppy$ — больше сообщений о слишком длинных строках вы, скорее всего, не увидите. Разумеется, просто так ничего не бывает. В режиме, включаемом этой командой, в $\text{\LaTeX}'e 2.09$ допустимы сколь угодно жидкие строки (\TeX по-прежнему старается, чтобы их было поменьше и чтобы они были не очень жидкими, но если надо выбирать между переполнением и жидкой строкой, он обязательно выберет второе). Как будет выглядеть при этом абзац, заранее предсказать трудно. Какие-то абзацы могут смотреться приемлемо, какие-то — чудовищно. Поэтому ставить $\backslash sloppy$ в преамбулу, задавая тем самым этот режим на весь текст, несколько рискованно. Разумнее задать эту команду перед концом того абзаца, в котором произошел *overfull*, и посмотреть, что из этого получится.

- ε В $\text{\LaTeX}'e 2\varepsilon$ команда $\backslash sloppy$ действует более аккуратно; если поставить ее в преамбулу, то в девяноста процентах случаев грубые недостатки, связанные с переполнениями и жидкими строками, пропадают (в отдельных случаях может все-таки возникнуть *overfull*). Читателю, не имеющему квалификации технического редактора, рекомендуется для первого раза принять к сведению этот рецепт и не вникать в последующие детали, связанные с $\backslash tolerance$ и $\backslash emergencystretch$.

Если вы не хотите, чтобы действие $\backslash sloppy$ распространялось и дальше, надо либо вернуться в обычный режим с помощью команды $\backslash fussy$, либо давать команду $\backslash sloppy$ внутри группы, с тем, чтобы этот режим кончился по выходе из группы. В любом случае необходимо понимать, на какие участки текста распространяется действие команд типа $\backslash sloppy$, влияющих на верстку абзаца. Правило таково:

Режим верстки абзаца определяется в тот момент, когда транслятор $\text{\TeX}'a$ читает пустую строку, завершающую абзац.

В частности, если вы решили дать команду `\sloppy` внутри группы, то для того чтобы она подействовала на абзац, необходимо, чтобы закрывающая фигурная скобка шла *после* пустой строки, завершающей абзац. Вот пример того, как надо действовать в таких случаях:

Черепаховый суп — изысканное деликатесное и диетическое блюдо

Черепаховый суп --- изысканное деликатесное и диетическое блюдо
`{\sloppy}`

}

А вот — типичная ошибка начинающего:

Черепаховый суп — изысканное деликатесное и диетическое блюдо

`{\sloppy}`
 Черепаховый суп ---
 изысканное деликатесное
 и диетическое блюдо}

Группа завершилась до пустой строки, поэтому к моменту, когда \TeX , увидев эту пустую строку, получил команду «сверстать абзац», он снова был в стандартном режиме, и вместо желаемых жидких, но не выходящих за край строк произошло переполнение (в первой строке).

Существует также способ не концентрировать всю разреженность в одной строке, как может получиться в результате действия команды `\sloppy`, а распределить ее более или менее равномерно по всему абзацу. Для этих целей используется параметр `\emergencystretch`. Это — некоторая длина, по умолчанию равная нулю. Ниже мы объясним точный смысл этого параметра, но для первого раза достаточно запомнить, что, если установить его значение равным примерно 3–5 пунктам, т. е. написать, например,

`\emergencystretch=5pt`

то в случае, когда без переполнений сверстать абзац не удается, \TeX попробует сделать все строки абзаца более разреженными (тем более разреженными, чем больше величина этого параметра). Точную величину `\emergencystretch` надо подбирать экспериментально.

7.3. Ручное управление разрывами строк

Иногда возникает необходимость повлиять на то, в каких местах \TeX начинает новую строку при верстке абзаца. Для этой цели есть соответствующие команды, и с одной из них мы уже встречались: это «неразрывный пробел», позволяющий запретить разрыв строки между двумя словами.

Иногда надо обеспечить, чтобы в каком-то слове не делалось переносов, причем не вообще никогда (тогда разумно применить команду `\hyphenation`), а только в данном месте. Для таких целей удобно применить команду `\mbox` следующим, например, образом:

Параметр `filename` задает имя файла. Параметр `\mbox{\bf filename}` задает имя файла.

Команда `\mbox` имеет один обязательный аргумент: в фигурных скобках может находиться любой текст (в том числе, как вы заметили, с командами переключения шрифта и т. п.); \TeX будет рассматривать содержимое `\mbox`'а как одну большую букву, и тем самым, конечно, не сможет разорвать его между строками.

Вы уже встречались с командой `\mbox`, если прочли в предыдущей главе раздел о включении текста в формулы; более подробно мы ее рассмотрим в главе о «блоках».

Есть еще один, несколько хулиганский, способ запретить \TeX 'у делать перенос в данном слове и в данном месте: надо в конце слова без пробела поставить команду `\-`, например, так:

Слово корова\`- в данном своем вхождении
перенесено не будет.

Автор позаимствовал это прием из книги [6].

Теперь посмотрим, что делать, если вам понадобилось насилино разорвать строку в каком-то месте, не начиная при этом нового абзаца. Для этого есть несколько способов, в зависимости от того, что вы хотите получить. Один из вариантов — воспользоваться командой `\|` и получить возможно не доходящую до края, но не растянутую строку:

Эта строка
была разорвана. Справа осталось
пустое место, но зато строка не
разреженная.

Эта строка\| была разорвана.
Справа осталось пустое
место, но зато строка
не разреженная.

Можно также воспользоваться командой `\linebreak`; при этом оборванная строка будет выровнена по правому краю, даже если ради этого ее придется растянуть:

Эта строка была
разорвана. Она выровнена по
правому краю, но для этого ее
пришлось безбожно растянуть.

Эта строка была\linebreak
разорвана. Она выровнена по
правому краю, но для
этого ее пришлось
безбожно растянуть.

Если строка действительно окажется растянутой, то вы к тому же получите и сообщение об этом во время трансляции. Если абзац длинный, а команда `\linebreak` расположена не слишком близко к его началу или концу, то скорее всего никаких растянутых строк не будет.

Команда `\ \` допускает и необязательный аргумент (см. с. 23): если в квадратных скобках указать какое-то расстояние (в ТЕХ'овских единицах длины — с. 25), то после оборванной строки будет оставлено это расстояние (по вертикали). Пример:

`Разорвем строку
и оставим место.`

`Разорвем строку\\[5pt] и
оставим место.`

При использовании команды `\ \` с необязательным аргументом бывает удобно вместо расстояния в явном виде указать один из следующих параметров:

`\smallskipamount` Маленький вертикальный пробел;

`\medskipamount` Вертикальный пробел побольше

`\bigskipamount` Еще больше.

Точный размер этого пробела зависит от стиля документа; на с. 121 изображена величина соответствующих пробелов в стандартных стилях со шрифтом кегля 11.

Команда `\ \` имеет и вариант со звездочкой (см. разд. I.2.9); если бы мы написали `*` или `*[расстояние]`, то эффект был бы тот же, что и без звездочки, и к тому же было бы запрещено заканчивать страницу на оборванной строке.

У команды `\linebreak` также может присутствовать необязательный аргумент. При этом команда `\linebreak[n]` указывает, что в данном месте желателен переход на новую строку, причем `n` указывает « силу » этого желания (`n` может быть целым числом от 0 до 4). Если `n = 4`, то это полностью равносильно `\linebreak` без необязательного аргумента, если `n = 0`, то это означает только, что строку в данном месте разрешается разорвать (так что применять эту команду с аргументом 0 между словами обычно бессмысленно); когда `n` возрастает от 1 до 3, команда `\linebreak[n]` « усиливает давление » на ТЕХ'овский алгоритм верстки абзаца, делая для него разрыв в указанном месте все более выгодным, невзирая на возможное появление жидких строк.

Есть также команда `\nolinebreak`, действующая противоположно; она также может принимать необязательный аргумент — целое число от 0 до 4. Будучи заданной без аргумента, эта команда запрещает разрыв строки в указанном месте; точно так же она действует, если ее необязательный аргумент равен 4. Когда ее необязательный аргумент возрастает от 1 до 3, ТЕХ'

начинает рассматривать разрыв строки в указанном месте как все менее желательный, даже невзирая на то, что из-за отказа от этого разрыва могут появиться жидкие строки.

Для простых приложений, о которых идет речь в этой главе, команды `\linebreak` и `\nolinebreak`, как правило, не нужны: чтобы разорвать строку, не растягивая, нужна команда `\``, а для запрещения разрыва гораздо удобнее «неразрывный пробел». По-настоящему эти две команды требуются только при разработке собственных макроопределений, о чем сейчас говорить преждевременно.

7.4. Верстка абзацев без выравнивания и переносов

Можно перевести \TeX в режим, при котором он вообще не будет пытаться выравнивать текст по правому краю и не будет делать переносов. Для этого служит команда `\raggedright`. Ее можно дать как в преамбуле, так и внутри документа; в любом случае, чтобы она подействовала на абзац, необходимо, чтобы ее действие не прекратилось до того, как \TeX 'ом будет прочтена пустая строка, завершающая абзац (ср. выше обсуждение команды `\sloppy`). Вот пример:

Этот абзац мы сверстали без
выравнивания и переносов.
Может быть, вид и не очень
аккуратный, зато без overfull'ов.

Этот абзац мы сверстали
без выравнивания и переносов.
Может быть, вид и не очень
аккуратный, зато без
overfull'ов. {\raggedright

}

Команда `\raggedright` в том виде, как она представлена в \LaTeX 'е, делает абзацный отступ равным нулю, поскольку предназначена для оформления текста в виде так называемого «флагового набора». В приведенном выше примере этого не произошло, поскольку команда `\raggedright` была выполнена после начала абзаца, когда абзацный отступ уже был определен; если, однако, записать ее в преамбулу, то отступ будет равен нулю для всех абзацев. Если вам это не нравится, но выравнивать текст по правому краю все-таки не хочется, можно после `\raggedright` записать в преамбуле команду, устанавливающую значение абзацного отступа `\parindent` (см. с. 22; в стандартных стилях значение этого параметра равно примерно 1.5 em).

7.5. Более тонкая настройка

Режимы, задаваемые командами `\sloppy` и `\fussy`, представляют собой две крайности. Здесь мы расскажем вам о более аккуратных способах

управления версткой абзацев. При первом чтении этот раздел можно пропустить.

Параметр `\hfuzz`. Если вы получаете слишком много сообщений о переполнениях, можно попросить TeX вообще не считать слишком длинными те строки, которые выдаются за край не очень сильно. Для этих целей предусмотрен параметр `\hfuzz`. Например, команда

`\hfuzz=2.5pt`

указывает, что как `overfull` будут восприниматься лишь те строки, которые выступают за край более, чем на два с половиной пункта. В обычном режиме значение параметра `\hfuzz` равно одной десятой пункта.

На первый взгляд такой способ борьбы с переполнениями напоминает страусову политику: вместо того, чтобы преодолевать трудность, мы делаем вид, что ее не существует. Тем не менее, как показывает опыт, если `\hfuzz` равен примерно 0.5 пункта, то получается приемлемый для ординарных изданий результат. Дело в том, что на фоне идеально выровненных абзацев одна выдающаяся на 0.5 пункта строка смотрится хуже, чем длинный текст, где все абзацы выровнены не идеально, а «с точностью до 0.5 пункта».

Мера разреженности строки. Как вы помните, в сообщении TeX'a о разреженной строке фигурирует такая мера разреженности строки, как `badness`. Посмотрите, как выглядят на печати разреженные строки с различными значениями этой меры:

Как может выглядеть разреженная строка.	<code>badness = 0</code>
Как может выглядеть разреженная строка.	<code>badness = 187</code>
Как может выглядеть разреженная строка.	<code>badness = 1215</code>
Как может выглядеть разреженная строка.	<code>badness = 2050</code>
Как может выглядеть разреженная строка.	<code>badness = 3907</code>
Как может выглядеть разреженная строка.	<code>badness = 10000</code>

У последней из наших строк значение `badness` равно 10000. Если растянуть пробелы в строке еще сильнее, то `badness` уже не увеличится, а останется равной 10000: с точки зрения TeX'a, такие разреженные строки настолько плохи, что нет смысла делать различия между ними.

Для интересующихся объясним подробнее, как вычисляется `badness`. Как мы уже говорили в разд. 2.2, промежутки между словами в тексте не фиксированы, а могут растягиваться или сжиматься. Каковы эти промежутки и насколько они могут растягиваться, зависит от шрифта (для примера: у основного шрифта кегля 10 обычный промежуток между словами равен примерно

3.33 пункта, и при этом он может растягиваться на 1.67 пунктов; промежуток же между предложениями в этом шрифте равен 4.44 пункта и может растягиваться на 5 пунктов)¹². Когда TeX растягивает строку с целью выравнивания, он находит сумму «пределов растяжимости» всех промежутков — это «предел растяжимости» строки, — и вычисляет, насколько требуемая длина строки больше «естественной» (определенной размерами слов и нерастянутых промежутков между словами) — это «требуемое растяжение» строки. Отношение «требуемого растяжения» к «пределу растяжимости» строки определяет, насколько разреженной получится строка. Традиционно это отношение обозначается буквой r . Практически в качестве меры разреженности используется не само число r , а число $100r^3$ — это и есть *badness*. Если даже окажется, что $100r^3 > 10000$, *badness* все равно будет считаться равной 10000: строки, для которых отношение r больше или равно 4.7 (примерно при этом значении получается 10000), рассматриваются TeX'ом как одинаково плохие.

В том счастливом случае, когда требуемая длина строки совпадает с естественной, мера разреженности равна нулю; если мера разреженности не пре- восходит 100, то растяжение строки не превосходило предела; на самом деле даже строки, мера разреженности которых не превосходит 200, выглядят все еще хорошо, хотя они уже и рассматриваются TeX'ом как слегка разреженные. TeX старается, чтобы такая «слегка разреженная» строка не попалась в абзаце рядом со строкой, в которой промежутки между словами сжимались.

Теперь мы можем объяснить точный смысл параметра `\emergencystretch`. Если при верстке абзаца не удалось избежать переполнения, то, при условии, что значение `\emergencystretch` отлично от нуля, TeX делает еще одну попытку верстки, при которой в процессе перебора вариантов разбиения абзаца на строки (и вычислений соответствующих значений *badness*) к «пределу растяжимости» каждой из строк прибавляется значение `\emergencystretch`.

Параметр `\tolerance`. Теперь в нашем распоряжении есть все необходимые понятия, чтобы объяснить, как TeX выбирает между разреженной строкой и переполнением (см. с. 104)

При верстке абзаца TeX никогда не создает строки, мера разреженности которых больше, чем значение TeX'овского параметра, называемого `\tolerance`. При невозможности удовлетворить этому условию создаются строки, выходящие за край: возникает *overfull*. С другой стороны, если мера разреженности строки не превосходит значения `\tolerance`, то будет создана именно столь разреженная строка, но не *overfull*.

В отличие от некоторых других систем компьютерной верстки, TeX никогда не растягивает и не сжимает отдельное слово.

В стандартном режиме значение параметра `\tolerance` равно 200. В L^AT_EX'e 2.09 действие команды `\sloppy` сводится в основном к тому,

¹²Кроме того, промежутки могут и сжиматься; пока речь идет только о жидких строках, это несущественно.

что она устанавливает значение `\tolerance` равным 10000, т. е. максимально возможному. Это объясняет, почему в режиме, определенном этой командой, \TeX создает сколь угодно разреженные строки. При этом, так как \TeX не различает строки с мерой разреженности, большей 10000, может получиться так, что одна из строк абзаца окажется совершенно ужасной: \TeX вложит в нее «всю разреженность», чтобы не увеличивать именно то число, которое \TeX минимизирует при переборе различных вариантов верстки данного абзаца (грубо говоря, это число тем больше, чем больше разреженных строк). Поэтому разумным решением во многих случаях будет увеличить значение `\tolerance`, но не до максимума, как это делает команда `\sloppy`, а до более разумной величины (скажем, 300 или 400). После этого \TeX , с одной стороны, получит большую свободу при верстке абзаца, а с другой — не сможет уже создавать абзацы, в которых все строки, кроме одной, приемлемы, а одна разрежена до безобразия.

- ε В $\text{\LaTeX}'$ е 2_ε команда `\sloppy` устанавливает $\tolerance = 9999$, а не 10000 (так что сколь угодно разреженные строки все-таки не допускаются), и при этом задает значение `\emergencystretch`, равное 3 em (так что при необходимости растянуть строки \TeX может равномерно распределить дополнительную растяжимость по всему абзацу). Не удивительно, что такая команда работает ε лучше, чем ее тезка в $\text{\LaTeX}'$ е 2.09

Увеличить значение `\tolerance` можно «глобально», во всем документе, дав в преамбуле команду наподобие

`\tolerance=400`

или же «локально», дав аналогичную команду внутри группы, содержащей данный абзац. В последнем случае не забывайте, что закрывающая группу фигурная скобка должна идти после пустой строки, завершающей абзац (см. выше обсуждение команд `\sloppy` и `\raggedright`).

Как менять длину абзаца. Иногда абзац не помещается на страницу из-за того, что он на строку-другую длиннее, чем нужно. В этом случае можно попросить \TeX сделать его короче на одну или две строки. Для этого надо написать

`\looseness=-1`

— и \TeX будет стараться, чтобы абзацы занимали на одну строку меньше, чем при оптимальной верстке. Если абзац короткий (скажем, занимает всего две строки), то из этого, конечно, ничего не получится. Если же абзац достаточно длинный, то у $\text{\TeX}'$ овского алгоритма верстки обычно хватает гибкости, чтобы достигнуть этой цели.

Можно присвоить параметру `\looseness` и значение `-2`; в этом случае TeX будет стараться делать абзацы короче на две строки (если не выйдет, то хоть на одну, а если и это не выходит, то оставит все как есть). Можно также присваивать `\looseness` положительные значения — в этом случае TeX будет стараться делать абзацы, которые содержат *больше* строк, нежели оптимальные.

По умолчанию значение параметра `\looseness` равно, естественно, нулю, и по окончании верстки каждого абзаца этот параметр также устанавливается в нуль. Тем самым нет нужды заботиться о том, чтобы значение `\looseness` менялось внутри группы, и бессмысленно присваивать этому параметру какое-то значение в преамбуле (оно забудется после первого же абзаца текста). Для каждого абзаца, для которого это вообще нужно, значение `\looseness` надо устанавливать заново.

Дополнительные тонкости с переносами. Вы можете влиять на частоту переносов в абзацах, сверстанных TeX'ом. Для этой цели предназначен параметр `\hyphenpenalty`. По умолчанию его значение равно `50`. Если присвоить этому параметру большее значение, то переносов будет меньше. Точнее говоря, если у TeX'a будет возможность выбирать, сделать лишний перенос или же обойтись без него, растянув строку чуть больше,¹³ то TeX будет склоняться ко второй из этих альтернатив тем чаще, чем большее значение `\hyphenpenalty`. Максимально возможное значение параметра `\hyphenpenalty` равно `10000`. Если в момент верстки абзаца это значение именно таково, то переносы в этом абзаце будут вообще запрещены. Такой режим верстки разумно, например, установить для абзацев, написанных на языке, для которого в вашей реализации TeX'a нет таблицы переносов, чтобы TeX не сделал переносов во французском тексте по английским правилам.

Наряду с `\hyphenpenalty` (отвечающим как за автоматически вставленные переносы, так и за переносы, возможные места для которых вы отметили с помощью команды `\-`), есть и параметр `\exhyphenpenalty`, отвечающий за переносы в словах с дефисом. Напомним, что в таких словах автоматический перенос возможен только в том месте, где дефис делит слово на части. Так вот, чем большее значение `\exhyphenpenalty`, тем с меньшей охотой TeX будет делать переносы в этих местах. Если же значение `\exhyphenpenalty` равно `10000`, то такие переносы будут и вовсе запрещены.

Значение двух описанных выше параметров используется TeX'ом в тот момент, когда он видит пустую строку, завершающую абзац. Соответственно, если вы присваиваете этим параметрам новые значения

¹³Не превышая значения `\tolerance`, разумеется!

внутри группы, то группа не должна завершаться до этой пустой строки. Учтите также, что, если вы увеличиваете значение `\hyphenpenalty` и тем самым затрудняете \TeX 'у переносы слов, то вам может понадобиться увеличить и `\tolerance`, чтобы он смог побольше растягивать строки.

Наконец, вот заключительная хитрость. Если вы присвоите значение 0 параметру `\uchyph`, написав

`\uchyph=0`

то \TeX никогда не будет делать переносов в словах, начинающихся с большой буквы. Такой режим полезен, например, в том случае, если вы не хотите делать переносы в именах собственных. Чтобы снова разрешить \TeX 'у переносить слова, начинающиеся с заглавной буквы, присвойте параметру `\uchyph` значение 1.

8. Между абзацами

В предыдущих разделах мы обсуждали, что происходит с документом «на уровне строки». Теперь изменим масштаб наших рассмотрений: будем смотреть не только на строки и абзацы, но и на то, как они расположены на странице.

8.1. Понятие о режимах \TeX 'а

В процессе обработки исходного текста \TeX в каждый момент находится в одном из трех режимов: горизонтальном, вертикальном или математическом. Несколько упрощая ситуацию, дело можно представить так:

- В процессе обработки текста (от появления первой же буквы до команды «закончить абзац», например, пустой строки) \TeX находится в горизонтальном режиме.
- Между абзацами, а также в начале работы (например, в процессе обработки преамбулы к \LaTeX овскому файлу) \TeX находится в вертикальном режиме.
- При обработке математических формул (см. гл. II) \TeX находится в математическом режиме.

В вертикальном режиме все пробелы и пустые строки игнорируются, так что между пустой строкой, завершающей абзац, и новой порцией

собственно текста незачем заботиться о лишних или недостающих пробелах (ср. с. 18).

В качестве команды «закончить абзац» можно использовать, наряду с известной вам пустой строкой, команду `\rag`.

Это — абзац, который мы не намерены завершать пустой строкой, как раньше.

А это уже совсем другой абзац.

Это --- абзац, который мы не намерены завершать пустой строкой, как раньше. `\rag` А это уже совсем другой абзац.

Иногда для ясности, когда в исходном файле присутствует сложная комбинация из TeX'овских команд, имеет смысл обозначить конец абзаца именно таким способом.

Идущие подряд несколько команд `\rag`, команда `\rag`, за или перед которой следует пустая строка, и т. п. — все это равносильно одной пустой строке или одной команде `\rag` (точно так же, как несколько пустых строк равносильны одной); дополнительный промежуток между абзацами вы таким образом не создадите. В разд. 8.4 рассказано, как получить на печати дополнительные вертикальные промежутки.

Сказанное в предыдущем абзаце можно с помощью понятия режима сформулировать так: в вертикальном режиме команда `\rag` ничего не делает.

8.2. Подавление абзацного отступа

Иногда возникает необходимость создать абзац, в котором нет абзацного отступа. Для этой цели удобно воспользоваться командой `\noindent`. В том абзаце, отступ в котором вы хотите подавить, эта команда должна идти первой (до любого текста):

Этот абзац будет сверстан без отступа.

В этом абзаце отступ будет присутствовать.

`\noindent` Этот абзац будет сверстан без отступа.

В этом абзаце отступ будет `\noindent` присутствовать.

Команда `\noindent` действует только на тот абзац, который с нее начинается; если ее поместить внутри абзаца, то вообще ничего не произойдет (что и иллюстрирует второй из абзацев в нашем примере). Стало

быть, между `\noindent` и абзацем, к которому она относится, не должно быть пустой строки (иначе получится, что `\noindent` относится к «пустому абзацу», заканчивающемуся этой пустой строкой).

В большинстве случаев, когда разумно сделать абзац без отступа, \LaTeX заботится об этом сам, так что вам не придется пользоваться командой `\noindent` чрезесчур часто.

Пользуясь понятием режима, можно сказать так: в вертикальном режиме команда `\noindent` означает «начать новый абзац без абзацного отступа», а в горизонтальном (и математическом, коль на то пошло) режиме она означает «ничего не делать».

8.3. Управление разрывами страниц

Как вы могли убедиться из раздела, посвященного абзацам, \TeX предоставляет широкие возможности для управления видом абзаца, местами разрывов строк и т. п. С разрывами страниц все обстоит не столь хорошо. Дело в том, что при верстке абзаца \TeX сначала читает его целиком, а затем перебирает различные способы разбиения на строки и выбирает из них оптимальный. При разбиении на страницы такой подход невозможен: если читать сразу весь текст, а затем перебирать различные варианты разбиения его на страницы, то компьютеру не хватит памяти. Поэтому разбиение на страницы в $\text{\TeX}'e$ — процесс «одноразовый». Сверстав очередной абзац, \TeX проверяет, набралось ли уже достаточно строк, чтобы заполнить страницу. Если оказывается, что достаточно, он производит разрыв страницы, и при этом выбор обычно невелик (часто бывает возможно сместить место разрыва страницы на строчку-другую за счет того, что некоторые интервалы между строками можно слегка растягивать или сжимать; таковы обычно интервалы между абзацами, между текстом и выключными формулами, но не между строками внутри абзаца). Имея все это в виду, рассмотрим, какие команды предоставляет \TeX для управления разрывами страниц.

Запрет разрыва страницы. Чтобы запретить разрыв страницы, используется команда `\nopagebreak`. Если поставить ее после конца абзаца, то разрыв страницы после этого абзаца будет запрещен. Если после конца абзаца присутствуют совместно как команда `\nopagebreak`, так и команда для дополнительных вертикальных промежутков, то команда `\nopagebreak` должна идти первой, в противном случае она не подействует.

Команда `\nopagebreak` может принимать необязательный аргумент — целое число от 0 до 4. Будучи снабжена этим аргументом, она не запрещает разрыв страницы в указанном месте, но делает его менее

выгодным с точки зрения ТЕХ'а (тем менее выгодным, чем больше аргумент). Команда `\nopagebreak[4]` означает полный запрет разрыва, как если бы команда была дана вообще без аргумента. Если аргумент равен 0, это означает только, что в данном месте страницу в принципе можно разорвать.

Наряду с командами, запрещающими разрыв страниц в указанном месте, ИАТЕХ предоставляет способ «глобально» затруднить ТЕХ'у разрывы страниц. Для этого служит команда `\samerpage`. После этой команды разрывы страниц станут возможны только между абзацами, но не внутри абзацев и не между текстом и выключной формулой. Если дать команду `\samerpage` внутри группы, то после конца группы действие этой команды прекращается (потому что это действие сводится к изменению некоторых не рассмотренных нами параметров).

Принудительный разрыв страницы. Для принудительного разрыва страниц в ИАТЕХ'е существует несколько способов. Первый и самый простой — команда `\newpage`. Под действием этой команды текущая страница завершается и дополняется снизу пустым пространством, если высота страницы получается меньше, чем надо.

Команда `\clearpage` также предназначена для принудительного разрыва страницы. Если пользоваться только теми средствами ИАТЕХ'а, которые были описаны до этого момента в нашей книге, то она будет работать в точности так же, как `\newpage`. В том же случае, если к моменту подачи этой команды остались так называемые «плавающие» иллюстрации или таблицы (см. разд. IV.8), то перед выдачей новой страницы они будут напечатаны.

Команда `\cleardoublepage` делает то же, что и `\clearpage`, но при этом в некоторых стилях (в тех, которые предусматривают разные поля для страниц с четным и нечетным номером — см. в разд. IV.1 по поводу стилевой опции `twoside`) новая страница обязательно имеет нечетный номер (если необходимо, при этом создается дополнительная пустая страница).

Все перечисленные команды для разрыва страницы действуют даже в том случае, если команда `\samerpage` ранее его запретила.

Если поставить подряд две команды `\newpage` (или `\clearpage`), то в печатном тексте чистая страница не получится. Чтобы создать чистую страницу, надо ИАТЕХ немного «обмануть»: в промежутке между двумя командами для разрыва страницы дать команду `\mbox{}`.

Наконец, существует команда `\pagebreak`, формально аналогичная команде `\linebreak` (см. с. 109). Если дать ее без аргументов, то страница в этом месте будет разорвана; при этом не исключено, что будет

сделана попытка выровнять ее по высоте с остальными страницами за счет растяжения тех вертикальных интервалов, которые можно растянуть (как правило, это интервалы между абзацами). Разумеется, вид у такой страницы еще хуже, чем у «укороченной» страницы с нормальными интервалами. Если дать команду `\pagebreak` с необязательным аргументом (целым числом от 0 до 4), то этот аргумент будет выражать степень желательности разрыва страницы в данном месте: если 0, то это всего лишь разрешение разорвать страницу, если 4, то разрыв обязательен, в остальных случаях степень желательности растет с ростом аргумента от 1 до 3.

Каждую из названных команд можно дать не только между абзацами, но и внутри абзаца; при этом разрыв страницы произойдет (или будет запрещен) после той строки, в которую попадает текст, соседствующий с этой командой.

Изменение высоты отдельной страницы (Л^AT_EX 2 _{ϵ}) Иногда при окончательной отделке текста бывает необходимо немного увеличить или уменьшить размер отдельно взятой страницы (чтобы, например, втиснуть в нее еще одну строку, которая иначе окажется в одиночестве на следующей полосе). Л^AT_EX 2 _{ϵ} предоставляет для этого следующее средство: на той странице, размер которой надо увеличить на одну строку, поместить между абзацами команду

`\enlargethispage{\baselineskip}`

Если надо увеличить размер на две строки, а не на одну, напишите в фигурных скобках `2\baselineskip` вместо `\baselineskip`; можно также в аргументе команды `\enlargethispage` написать `-\baselineskip`, `-2\baselineskip`, и т. п. В этом случае высота полосы уменьшится на одну, две и т. д. строки.

Добавим несколько Т_EX'нических подробностей. Во-первых, если текст набирается в две колонки, то команда `\enlargethispage` действует только на одну из них — на ту, в которую она попала. Во-вторых, при действии команды `\enlargethispage` увеличенная полоса может заехать на строку с колонцифой, если таковая предусмотрена стилем документа. И наконец, в аргументе команды `\enlargethispage` может стоять не только кратное `\baselineskip`, но и любая длина, выраженная в Т_EX'овских единицах (скажем, 5mm), хотя смысла в указании такого аргумента этой команды, как правило, нет.

Подробности о параметре `\baselineskip` и колонцифре — в следующей главе. О двухколонном наборе будет рассказано в разд. III.8.6.

8.4. Вертикальные промежутки

Большинство вертикальных промежутков (например, между заголовком раздела и его текстом) **LATEX** устанавливает самостоятельно, и вам об этом можно не заботиться. Иногда возникает необходимость сделать дополнительный вертикальный промежуток между абзацами. Подобно тому, как внутри абзацев для задания промежутков вручную разумнее пользоваться не командами, явно задающими размер промежутка, а командами вроде `\,`, или `\quad`, так и для задания промежутков между абзацами в первую очередь полезны такие команды:

- `\smallskip` задает такой — промежуток;
- `\medskip` задает такой — промежуток;
- `\bigskip` задает такой — промежуток.

Проще всего поставить эти команды непосредственно после пустой строки или команды `\rag`, завершающей абзац:

После этого абзаца мы оставим дополнительный пробел.

А теперь начнем новый абзац.

После этого абзаца мы оставим дополнительный пробел.

`\par\smallskip`

А теперь начнем новый абзац.

Конкретная величина промежутков, задаваемых перечисленными командами, зависит от стиля документа. Эти размеры совпадают со значениями параметров `\smallskipamount`... `\bigskipamount`, о которых шла речь на с. 110.

Если вы хотите задать размер вертикального промежутка в явном виде, можно воспользоваться командой `\vspace`. Подобно команде `\hspace` (см. с. 91), у нее есть один обязательный аргумент — величина промежутка. Например, можно написать

`\vspace{2ex}`

Команду `\vspace` удобнее всего ставить после конца абзаца (подобно таким командам, как `\smallskip`).

Можно поставить команду `\vspace` (или `\smallskip` и т. п.) не после пустой строки или `\rag`, а непосредственно перед ними, после всего текста абзаца. Если поставить какую-либо из этих команд внутри абзаца, то дополнительный вертикальный пробел получится не между абзацами, а между строками абзаца.

Если дать команду `\vspace` сразу же после `\newpage` или `\clearpage`, то вертикального отступа в начале новой страницы не получится; вертикальный отступ, создаваемый `\vspace`, пропадет и в том случае, если он оказывается в начале новой страницы, получившейся «естественному образом». Чтобы вертикальный отступ в начале страницы не пропадал, надо воспользоваться вариантом со звездочкой после имени команды: если написать `\vspace*{1cm}`, то будет создан вертикальный промежуток в 1 см, не пропадающий даже в том случае, если команда дана сразу после `\newpage` или `\clearpage` или в этом месте произошел разрыв страницы.

Можно заставить команду `\vspace` создать промежуток не фиксированной, а переменной длины. Именно, в самом общем виде эта команда записывается так:

```
\vspace{x plus y minus z}
```

Здесь *x*, *y* и *z* — длины, выраженные в $\text{\TeX'овских единицах}$, а *plus* и *minus* — так называемые «ключевые слова» $\text{\TeX}'а$ (в отличие от команд, перед ними не надо ставить backslash). При этом *x* обозначает «естественную» величину отступа: если при верстке страницы вертикальные интервалы не приходится растягивать или сжимать (если, например, мы разрешили $\text{\TeX}'у$ оставлять внизу страницы пустое место; в дальнейшем мы обсудим, как это делать), то будет сделан пробел размером ровно *x*. При необходимости, однако (например, ради того, чтобы все страницы имели одинаковую высоту), этот интервал можно будет и изменить: *y* указывает, насколько, самое большое, можно растянуть интервал, в то время как *z* указывает, насколько, самое большое, можно его ужать. Говоря $\text{\TeX}ническим языком$, команда `\vspace` вставляет в страницу «клей»¹⁴; расстояния, указанные после *plus* и *minus*, называются соответственно *plus* и *minus*-компонентами этого клея. Если *plus*- или *minus*-компонента в аргументе команды `\vspace` не указана, то соответствующий интервал не сможет растягиваться (сжиматься). Большинство вертикальных интервалов, автоматически вставляемых $\text{\LaTeX}'ом$, обладают растяжимостью и/или сжимаемостью, что помогает при нахождении оптимальных разрывов страниц.

Один частный случай растяжимых промежутков настолько важен, что в $\text{\LaTeX}'е$ для него предусмотрена специальная команда. Именно, в аргументе `\vspace` или `\vspace*` можно вместо длины, заданной в $\text{\TeX'овских единицах}$, написать `\fill`. Это задает промежуток нулевого размера, но обладающий способностью бесконечно растягиваться. Если, например, написать

¹⁴ Свойства которого мало похожи на свойства настоящего клея. См. гл. VIII.

```
\clearpage\vspace*{\fill}
\begin{center}
Заголовок
\end{center}
\vspace*{\fill}\clearpage
```

то слово «заголовок» будет расположено точно по центру отдельной страницы, созданной командами `\clearpage`.

Перед командой `\fill` в аргументе `\vspace` или `\vspace*` можно поставить коэффициент — целое число или десятичную дробь, и тогда растяжимость умножится на этот коэффициент. Например, если написать

```
\clearpage\vspace*{0.5\fill}
\begin{center}
Заголовок
\end{center}
\vspace*{\fill}\clearpage
```

то перед словом «заголовок» будет оставлено ровно в два раза меньше места, чем после него, так как `0.5\fill` растяжим в два раза меньше, чем `\fill`.

Теперь можно признаться, что горизонтальные промежутки, создаваемые командой `\hspace`, также могут быть растяжимыми; чтобы этого добиться, надо задать в аргументе команды `\hspace` не только «естественную длину», но еще и `plus-` и/или `minus`-компоненту. Например, если сказать

```
\hspace{1cm plus 2mm minus 1em}
```

то при верстке абзаца соответствующий интервал сможет растягиваться (самое большое — на 2 мм) или сжиматься (самое большое — на 1em). Можно также, вместо длин с `plus-` или `minus`-компонентами, написать `\fill` (возможно, с коэффициентом). В простых приложениях такие конструкции, как правило, не встречаются. Мы еще будем говорить о них в разд. VIII.3.3.

8.5. Интерлиньяж

В полиграфии этим красивым словом называется интервал между строками. \LaTeX 'овские команды наподобие `\small`, устанавливающие размер шрифта, автоматически устанавливают и размер интервала между строками, так что вручную менять его не следует (потому мы и не рассказываем, как это делать; любопытствующий читатель может узнать

подробности в книге [3]). Можно, однако (и иногда это бывает необходимо), пропорционально увеличить или уменьшить все интервалы между строками: например, для того чтобы подогнать число страниц в документе к требуемому. Если, скажем, вы хотите увеличить интервалы между строками на 1%, т. е. в 1.01 раза, то следует написать так:

```
\renewcommand{\baselinestretch}{1.01}
```

Вместо десятичной точки можно использовать и десятичную запятую. Если вы пользуетесь этой командой, помещайте ее в преамбулу.

Между абзацами можно организовать дополнительные вертикальные интервалы. Именно, в $\text{\TeX}'$ е есть параметр \parskip со значением длины; если присвоить ему ненулевое значение, например, написав

```
\parskip=3mm
```

то между абзацами будет делаться отступ в 3 мм (в дополнение к обычному межстрочному интервалу). Без особой необходимости не следует присваивать параметру \parskip новое значение, поскольку оно вполне разумно устанавливается в стандартных $\text{\LaTeX}'$ овских стилях.

На самом деле в стандартных стилях \parskip является *растяжимой* длиной (см. с. 122). Именно, естественный размер \parskip равен нулю, но у него есть еще *plus*-компонент, равная одному пункту. Стало быть, если вертикальные интервалы на странице не варьируются, то никакого дополнительного интервала между абзацами не делается, но если страницу при верстке приходится растягивать по вертикали, то каждый из интервалов между абзацами может быть растянут (максимум на один пункт). При желании можно изменить как естественный размер, так и растяжимую компоненту параметра \parskip с помощью команды \setlength , о которой пойдет речь в разд. VII.3. Надо полагать, что вы будете делать это сознательно.

8.6. Набор в две колонки

Если вам необходимо набирать в две колонки весь документ, то это надо сделать, указав в команде \documentstyle (\documentclass в $\text{\LaTeX}'$ е 2 ε) соответствующую «стилевую опцию» (см. разд. IV.1). Если же в две колонки надо набрать не весь текст, а только его часть, к вашим услугам команда \twocolumn . Действует она так: сначала выполняется команда \clearpage , а затем с новой страницы, созданной этой командой, начинается набор в две колонки.

Иногда бывает необходимо в начале новой страницы поместить один или несколько абзацев текста во всю ширину страницы, а оставшийся текст на этой странице набрать в две колонки. Для этих целей можно использовать команду \twocolumn с необязательным аргументом. Необязательный аргумент

(в квадратных скобках, как водится) — это тот текст, который будет напечатан во всю ширину страницы; если он состоит из нескольких абзацев, то абзацы, как обычно, разделяются пустыми строками.

Команда `\onecolumn` осуществляет переход от двухколонного набора к одноколонному (предварительно она опять-таки выполняет команду `\clearpage`).

8.7. Заключительные замечания о разрывах страниц и вертикальных интервалах

Мы уже отмечали, что TeX'овские алгоритмы создания страниц не обладают той же гибкостью, что алгоритм разбиения абзаца на строки. Поэтому не надо слишком увлекаться принудительными разрывами и запретами разрывов страниц и командами наподобие `\vspace*`. Даже такая замечательная программа, как TeX, не сможет удовлетворить логически противоречивым требованиям; если ограничений на разрывы страниц слишком много, то TeX будет вынужден сделать эти разрывы, исходя из формального смысла своих алгоритмов. При этом, скорее всего, на печати вы получите много страниц, разорванных в самых неподходящих и неудачных с точки зрения человека местах, а на экране — много сообщений вроде такого:

```
Underfull \vbox (badness 10000) has occurred
while \output is active
```

Если вы регулярно сталкиваетесь с такими неприятностями, имеет смысл заново продумать принципы организации вашего текста, а может быть, и изложить кое-что по-другому. Избавиться от растянутых по вертикали страниц можно, если дать в преамбуле документа команду `\raggedbottom`, разрешающую делать страницы неодинаковой высоты (некоторые стили дают эту команду автоматически, но даже если вы ее продублируете, ничего плохого не случится). Действие, противоположное `\raggedbottom`, вызывается командой `\flushbottom`.

Наконец, если трудности возникают оттого, что вы часто оставляете в тексте место командой `\vspace*` (например, чтобы вклейть рисунок), то вам стоит воспользоваться «плавающими» иллюстрациями (см. разд. IV.8).

9. Специальные абзацы

В разделе, посвященном абзацам, уже упоминалось о том, как можно верстать текст без выравнивания по правому краю. Сейчас речь пойдет о других подобных случаях, когда требуются абзацы специального

вида. Большинство описываемых в этом разделе способов верстки реализованы в виде окружений (см. с. 24); не забывайте, что всякое окружение ограничивает группу, так что если вам нужно будет не только получить текст специального вида, но и переключить шрифт, вы можете дать команду переключения шрифта внутри окружения и не заботиться специально о восстановлении прежнего шрифта: вместе с окружением кончится и группа, и прежний шрифт восстановится автоматически.

9.1. Цитаты

Если вам нужно включить в текст цитату, пример, предупреждение и т. п., то удобно воспользоваться окружением `quote`. Это окружение набирает текст, отодвинутый от краев. Пример:

Каждый владелец персонального компьютера должен понимать, что

не следует записывать на жесткий диск программы неизвестного назначения.

Поэтому подумайте, прежде чем переписывать программы у знакомых.

Как вы можете заметить из этого примера, текст, оформленный окружением `quote`, не имеет абзацного отступа и отделяется от окружающего текста пробелами. Если после `\end{quote}` дальнейший текст следует без пропуска строки, то на печати он начнется с новой строки, но без абзацного отступа (после включения цитаты продолжается прерванный абзац); если после `\end{quote}` пропустить строку, то после цитаты текст будет идти с абзачным отступом (если, разумеется, значение параметра `\parindent` не равно нулю — см. с. 22).

Для длинных цитат, состоящих из нескольких абзацев, удобнее использовать окружение `quotation`. Оно полностью аналогично `quote`, за тем исключением, что в тексте, оформленном этим окружением, делается абзацный отступ.

Каждый владелец персонального компьютера должен понимать, что

```
\begin{quote}
не следует записывать на
жесткий диск программы
неизвестного назначения.
\end{quote}
```

Поэтому подумайте, прежде чем переписывать программы у знакомых.

9.2. Центрирование, ровнение текста по краю

Для этих целей используются окружения `center` (для центрирования), а также `flushleft` и `flushright` (для верстки текста, выровненного по левому и правому краю соответственно).

Внутри каждого из этих окружений можно в принципе набирать и самый обычный текст, стандартным образом разбитый на абзацы с помощью пустых строк, но при этом каждая строка получающегося «абзаца» будет центрирована (для окружения `center`) или выровнена по левому или правому краю (для окружений `flushleft` и `flushright` соответственно).

Окружение `flushleft` по своему действию практически эквивалентно команде `\raggedright` (см. с. 111); но вообще, конечно, все перечисленные окружения нужны не для создания абзацев столь странного вида, а для организации текста в виде последовательности строк, каждая из которых центрирована или сдвинута влево или вправо; для этого достаточно после каждой строки, которую вы хотите центрировать (сдвинуть вправо ...), поставить команду `\`` (см. с. 109); следующая строка будет принадлежать тому же абзацу, и, стало быть, опять же будет центрирована (сдвинута). А если вы не будете делать разбиения на строки командой `\``, то это будет сделано автоматически, с появлением нескольких странных абзацев (на с. 24 приведен пример того, что может в этом случае получиться). Вот примеры разумного использования этих окружений:

наше дело правое	<code>\begin{flushright}</code> наше дело <code>\`</code> правое <code>\end{flushright}</code>
---------------------	---

левый марш	<code>\begin{flushleft}</code> левый <code>\`</code> марш <code>\end{flushleft}</code>
---------------	---

а мы всегда в центре	<code>\begin{center}</code> а <code>\`</code> мы <code>\`</code> всегда <code>\`</code> в центре <code>\end{center}</code>
-------------------------------	--

Как и в случае с `quote` и `quotation`, если после команды `\end`, закрывающей любое из этих окружений, в исходном тексте идет пустая строка, то следующий абзац набирается \LaTeX'om с обычным абзацным отступом, в противном случае — без отступа.

9.3. Стихи

В принципе стихи можно набирать с помощью окружений `flushleft` или `center`, разделяя строки командой `\backslash`, а строфы, например, пустой строкой и командой `\vspace{4pt}` (см. с. 110 и 121 по поводу этих команд). Кроме того, в `LATEX`'е для набора стихов предусмотрено специальное окружение `verse`. Строки в нем разделяются командой `\backslash`, а строфы — пустой строкой. При этом строки получаются выровненными по левому краю и отодвинутыми от левой границы текста. Если строка окажется слишком длинной, она будет перенесена на следующую строку (не исключено, что в каких-то словах будет сделан перенос) и сдвинута примерно на 15 пунктов вправо. Пример:

```
Здесь любит медведь
Иногда посидеть
И подумать: "А чем бы
    такое заняться?"
```

```
\begin{verse}
Здесь любит медведь\\
Иногда посидеть\\ И подумать:\\
"А чем бы такое заняться?"\\
\end{verse}
```

Создатель `LATEX`'а Лесли Лэмпорт предсказывал, что окружение `verse` будет обругано поэтами.

Наличие или отсутствие абзацного отступа в абзаце после окружения `verse` определяется по тем же правилам, что для `quote` и `quotation`.

9.4. Перечни

Для печати перечней используются окружения `itemize` (для простейших перечней), `enumitem` (для нумерованных перечней) и `description` (для перечней, в которых каждый пункт имеет заголовок — например, словарных статей или иных описаний). В любом случае элементы перечня вводятся командой `\item` (иногда — с необязательным аргументом). Разберем последовательно, как работают указанные окружения.

Простейшие перечни (`itemize`). Каждый элемент перечня вводится командой `\item` без аргумента.

- На печати каждый элемент перечня снабжается черным кружочком («горох» на жаргоне полиграфистов).
- Перечни могут быть вложенными друг в друга:
 - Максимальная глубина вложенности равна 4.
 - Отступы и символы перед элементами выбираются автоматически.

- На втором уровне элементы перечня отмечаются полужирными короткими тире, на третьем — звездочками, на четвертом — точками.
- При попытке вложить пять таких окружений \LaTeX выдаст сообщение об ошибке.

Вот как выглядел в исходном файле предшествующий текст:

```
\begin{itemize}
\item На печати каждый...
\item Перечни могут быть
вложенными друг в друга:
\begin{itemize}
\item Максимальная глубина вложенности равна 4.
\item Отступы и символы перед элементами
выбираются автоматически.
\end{itemize}
\item На втором уровне элементы...
\item При попытке вложить...
\end{itemize}
```

Внутри окружения `itemize` до первой команды `\item` не должно идти никакого текста или же команд, генерирующих текст. Если вы попытаетесь проигнорировать этот запрет, то \LaTeX выдаст вам сообщение об ошибке. Другие команды (например, команды смены шрифта) могут идти и до первого `\item`.

Окружение `itemize` можно использовать также для создания перечней, в которых каждый элемент имеет короткий заголовок. Для создания такого заголовка надо задать команде `\item` необязательный аргумент (в квадратных скобках, как водится). При наличии у этой команды необязательного аргумента стандартный значок, отмечающий элемент перечня («горошина», звездочка и т. п.) не печатается, а вместо него печатается текст, заданный в необязательном аргументе:

- Этот элемент перечня помечен стандартно.

`Раз` Здесь мы сами задали заголовок.

`Два` Здесь тоже.

```
\begin{itemize}
\item Э́тот элемент перечня
помечен стандартно.
\item[{\sf Раз}] Здесь мы сами
задали заголовок.
\item[Два] Здесь тоже.
\end{itemize}
```

Обратите внимание, что заголовки, заданные нами в необязательных аргументах команд `\item`, печатаются выровненными по правому краю, а также что команды смены шрифта в этих аргументах не распространяются на дальнейший текст.

Если заголовок, заданный вами в необязательном аргументе команды `\item`, будет слишком длинен, то он заедет на левое поле. В таких случаях, если вы хотите, чтобы заголовки элементов перечня были выровнены по левому, а не по правому краю, лучше пользоваться окружением `description`, о котором речь пойдет ниже.

Если первый отличный от пробела символ после команды `\item` является открывающей квадратной скобкой, то `LATEX` решит, что эта скобка начинает необязательный аргумент команды `\item`. Если при этом вы использовали эту скобку просто как типографский знак, то в результате получится сообщение об ошибке. Чтобы избежать такой неприятности, надо в этом случае квадратную скобку «спрятать», заключив ее в фигурные скобки:

`\item {[} --- редко встречающийся знак...`

Нумерованные перечни (`enumerate`). В таких перечнях каждый элемент также вводится командой `\item` без аргумента, но на печати он будет отмечен не значком, а номером (эти номера создаются `LATEX`'ом автоматически; если вы переставите какие-то элементы перечня, что-то добавите или удалите, нумерация автоматически изменится).

1. В окружении `enumerate` элементы списка нумеруются цифрами или буквами.
2. Нумерация производится автоматически.
3. Перечни могут быть вложенными друг в друга:
 - a) Максимальная глубина вложенности равна 4.
 - b) Отступы и обозначения для элементов выбираются автоматически.
4. На втором уровне элементы обозначаются строчными буквами, на третьем — римскими цифрами, на четвертом — прописными буквами.
5. При попытке вложить пять таких окружений `LATEX` выдаст сообщение об ошибке.

В исходном тексте это выглядело так:

```
\begin{enumerate}
\item В окружении {\tt enumerate}
элементы списка нумеруются
цифрами или буквами.
\item Нумерация производится автоматически.
\item Перечни могут быть
вложенными друг в друга:
\begin{enumerate}
\item Максимальная глубина
вложенности равна 4.
\item Отступы и обозначения для
элементов выбираются автоматически.
\end{enumerate}
\item На втором уровне элементы
обозначаются строчными буквами, на
третьем --- римскими цифрами, на
четвертом --- прописными буквами.
\item При попытке вложить пять
таких окружений \LaTeX{} выдаст
сообщение об ошибке.
\end{enumerate}
```

Внутри окружения `enumerate` до первой команды `\item` не должно идти никакого текста или же команд, генерирующих текст.

На номера элементов нумерованного перечня можно организовать автоматические ссылки с помощью команды `\ref` (см. с. 26). Делается это так.

Представим себе, что вам нужно сослаться на какой-то пункт нумерованного перечня (например, чтобы написать «Согласно пункту 3 настоящих Правил ... »). Если вы в ходе работы над текстом переставите какие-то пункты или добавите новые, то номер пункта 3 может измениться. Вместо того чтобы каждый раз отсчитывать, которым по счету идет этот пункт, можно пометить элемент перечня с помощью команды `\label` (см. разд. I.2.11). Команду `\label` удобно ставить сразу после команды `\item`, вводящей помечаемый элемент перечня, но в принципе можно поставить ее и позже — до следующего `\item`.

Ссылка на метку производится с помощью команды `\ref`. У нее также один обязательный аргумент — та самая метка, на которую вы хотите сослаться (ссылка на страницу, на которой расположена метка, производится, как обычно, с помощью команды `\pageref`). Пример:

1. Переходите улицу только на зеленый свет.
2. Стоящий трамвай обходить можно, а автобус — нет.

Согласно правилу 2, сформулированному на с. 132, обходить стоящий автобус нельзя.

```
\begin{enumerate}
\item Переходите улицу только на зеленый свет.
\item \label{tram} Стоящий трамвай обходить можно, а автобус --- нет.
\end{enumerate}
Согласно правилу \ref{tram}, сформулированному на с. \pageref{tram}, обходить стоящий автобус нельзя.
```

Символы неразрывного пробела мы поставили затем, чтобы номер правила или страницы не остался в одиночестве в начале строки.

В окружении `enumarate` команда `\item` также может иметь необязательный аргумент, который работает так же, как в окружении `itemize`. Соответственно, остается в силе и сделанное в предыдущем подпункте предупреждение относительно того, что будет, если первый отличный от пробела символ после `\item` является открывающей квадратной скобкой.

Перечни с заголовками (description). В этих перечнях каждый элемент, как уже было сказано, снабжен заголовком. Поэтому элементы перечня вводятся командой `\item` с необязательным аргументом (заключенным, стало быть, в квадратные скобки — см. с. 23), представляющим собой этот заголовок. Пример:

Летом можно собирать различные ягоды:

Летом можно собирать различные ягоды:

```
\begin{description}
\item[черника:] темно-синие, очень вкусные, хороши в свежем виде, варенье тоже получается хорошее;
\item[голубика:] синие, более водянистые, чем черника, и не такие вкусные;
\item[брусника:] ярко-красные, из них получается очень вкусное варенье.
\end{description}
```

черника: темно-синие, очень вкусные, хороши в свежем виде, варенье тоже получается хорошее;

черника: темно-синие, очень вкусные, хороши в свежем виде, варенье тоже получается хорошее;

голубика: синие, более водянистые, чем черника, и не такие вкусные;

голубика: синие, более водянистые, чем черника, и не такие вкусные;

брюслица: ярко-красные, из них получается очень вкусное варенье.

брюслица: ярко-красные, из них получается очень вкусное варенье.

Как вы могли заметить, заголовки элементов перечня оформляются в окружении `description` полужирным шрифтом. Если вас не устраивает этот шрифт, можно аргумент команды `\item` начать с команды переключения шрифта, скажем, `\rm` или `\sl`.

Внутри окружения `description` до первой команды `\item` не должно идти никакого текста или же команд, генерирующих текст.

Если в заголовке элемента перечня присутствует закрывающая квадратная скобка, то `\TeX` решит, что именно на ней заканчивается необязательный аргумент команды `\item`, в результате чего на печати получится совсем не то, что вы хотели. Чтобы избежать этой неприятности, надо эту квадратную скобку (либо, что еще проще, весь заголовок) заключить дополнительно в фигурные скобки (внутри квадратных).

Другие виды перечней. Если вас не устраивает стандартное оформление перечней (например, вид пометок, которыми отмечаются элементы перечня `itemize`), его несложно изменить. Как это сделать, будет рассказано в разд. VII.2.5. Несколько труднее, к сожалению, сделать так, чтобы буквы, которыми нумеруются элементы нумерованного перечня, были русскими, а не латинскими (как в примерах в этой книге). Желательно, чтобы в устанавливаемой у вас русификации `\TeX`'а были определены соответствующие команды. `\TeX` позволяет менять оформление перечней и более серьезным образом, создавая перечни иного типа, чем рассмотренные выше. На данный момент наш `\TeX`нический уровень не столь высок, чтобы можно было освоить эти возможности `\TeX`'а, но в гл. IX будет рассказано и об этом.

9.5. Буквальное воспроизведение (`verbatim`, `verb`)

Окружение `verbatim` предназначено для воспроизведения текста шрифтом `typewriter` в точности так, как он выглядит в файле. Одной только команды `\tt` для этого недостаточно, поскольку воспроизводимый текст может содержать, например, команды `\TeX`'а, и необходимо, чтобы они печатались, но не исполнялись.

Между `\begin{verbatim}` и `\end{verbatim}` могут идти любые символы, за исключением последовательности символов `\end{verbatim}` (в том числе, например, `\`` или не имеющие пар фигурные скобки). В строке `\end{verbatim}`, завершающей окружение `verbatim`, не должно быть пробела между `\end` и `{verbatim}` (вопреки общим правилам: обычно такой пробел, как и вообще пробел после имени команды, состоящего из букв, безвреден).

Короткие последовательности символов удобно набирать для буквального воспроизведения с помощью команды `\verb`. Непосредствен-

но после `\verb` должен стоять любой символ, не являющийся буквой или звездочкой, далее — воспроизводимый текст (укладывающийся в одну строку), *не* содержащий того символа, который стоял непосредственно после `\verb`, а затем — тот символ, что стоял непосредственно после `\verb`. После `\verb` не должно быть пробела. Пример:

Команда `\dots` задает многоточие. Знак " в \TeX 'е используется редко.

Команда `\verb"\dots"` задает многоточие. Знак "`\verb!"`" в \TeX 'е используется редко.

Описанные окружение и команда удобны, когда надо имитировать машинописный текст, текст на мониторе компьютера, или набирать тексты компьютерных программ. В данном руководстве `\verb` и `verbatim` широко использовались для набора \LaTeX 'овских и \TeX 'овских команд.

У команды `\verb` и окружения `verbatim` есть варианты «со звездочкой» (см. с. 25). От своих вариантов без звездочки они отличаются тем, что пробел изображается знаком `_`.

Команду `\verb` и окружение `verbatim` нельзя использовать в сносках; если вам необходимо напечатать в сноске что-нибудь вроде `\sqrt`, то придется делать это вручную:

```
{\tt \symbol{"5C}\sqrt{}}
```

Если вы забудете «закрывающий символ» в команде `\verb` или сделаете опечатку в тексте `\end{verbatim}`, то в лучшем случае получите уйму сообщений об ошибке, а в худшем — приведете \TeX в такое состояние, что компьютер придется перезагружать.

Если вы воспроизводите в режиме `verbatim` текст, простирающийся на многие страницы (например, компьютерную программу), то \TeX 'у может не хватить памяти. Чтобы избежать такой неприятности, надо распределить воспроизводимый текст по нескольким окружениям `verbatim`.

- ε Если вы пользуетесь \LaTeX 'ом 2ε, то можно подключить стилевой пакет `verbatim`, после чего вы сможете спокойно задавать сколь угодно длинные окружения `verbatim` и `verbatim*`. Только не забудьте про `\end{verbatim}` в конце.

9.6. Абзацы нестандартной формы

Пусть нам потребовалось создать абзац с «отрицательным» абзацным отступом, в котором все строки, кроме первой, начинаются на расстоянии 1 см от полей. Этого можно добиться следующим образом:

Отрицательный абзацный отступ (по-английски: hanging indentation).

`\hangindent=1cm \noindent`
Отрицательный абзацный отступ
(по-английски: hanging indentation).

Здесь TeX'овский параметр `\hangindent` означает величину отступа от полей во всех строках абзаца, кроме первой (по умолчанию значение этого параметра равно нулю). Обратите внимание, что мы начали абзац командой `\noindent`, чтобы первая строка не началась с абзацным отступом.

Пусть теперь нам хочется, чтобы дополнительный отступ, величина которого задана параметром `\hangindent`, начинался не со второй строки, а, скажем, с третьей. Для этого надо установить еще один TeX'овский параметр, обозначаемый `\hangafter`:

TeX позволяет сделать так, чтобы отступ начался не с первой строки, а там, где нам это потребовалось.

`\hangindent=1cm \hangafter=2`
`\noindent \TeX\` позволяет сделать так, чтобы отступ начался не с первой строки, а там, где нам это потребовалось.

Значение параметра `\hangafter` — номер строки, *после* которой начинается дополнительный отступ. По умолчанию значение `\hangafter` равно единице (как и было в нашем первом примере).

Можно также добиться того, чтобы дополнительный отступ не начинался *после* какой-то строки, а напротив, присутствовал только в нескольких первых строках абзаца. Для этого надо присвоить параметру `\hangafter` отрицательное значение: если величина `\hangafter` равна $n < 0$, то дополнительный отступ, равный `\hangindent`, будет присутствовать в строках номер $1, 2, \dots, |n|$. Пример:

С помощью рассмотренных нами средств TeX'a можно выкапывать в абзацах небольшие ямки. На место образующегося белого прямоугольника можно, например, вклейте иллюстрацию.

`\hangindent=1.5cm`
`\hangafter=-3 \noindent`
С помощью рассмотренных нами средств \TeX'a можно выкапывать в абзацах небольшие ямки. На место образующегося белого прямоугольника можно, например, вклейте иллюстрацию.

Можно сделать так, что дополнительный отступ, задаваемый параметром `\hangindent`, будет делаться не слева, а справа. Для этого значение этого параметра надо сделать отрицательным. Именно, если значение `\hangindent` равно $h < 0$, то дополнительный отступ размером $|h|$ будет отсчитываться от *правого*, а не левого поля (в каких

именно строках будет этот дополнительный отступ, по-прежнему определяется значением `\hangafter`):

На сей раз нам захотелось приклеить картинку не слева, а справа.

Что ж, TeX позволяет сделать и так, было бы желание. Вскоре вы сможете убедиться, что и это — не предел.

`\hangindent=-2cm \hangafter=2
\noindent`

На сей раз нам захотелось приклеить картинку не слева, а справа. Что ж, TeX\ позволяет сделать и так, было бы желание. Вскоре вы сможете убедиться, что и это ---- не предел.

После каждой команды «завершить абзац» (иными словами, после каждой пустой строки или команды `\par`) восстанавливаются принятые по умолчанию значения параметров `\hangindent` и `\hangafter`. Отметим еще, что было бы неразумно играть с этими параметрами внутри L^AT_EX'овских окружений наподобие `itemize` или `quote`.

Если вам не хватает возможностей, которые дает варьирование параметров

`\hangindent`

и `\hangafter`, то вот вам пример, как с помощью TeX'a создать абзац совсем уж причудливой формы. Все переносы в словах и места разрывов строк были найдены TeX'ом автоматически.

Начало этого причудливого абзаца выглядело в исходном тексте так:

```
\parshape=7
0cm 4cm 0.5cm 5cm 1cm 6cm 1.5cm 7cm
```

```
2cm 6.5cm 1.8cm 6cm 1.7cm 5cm
```

```
\noindent \small
```

Если вам не хватает возможностей...

Смысл этих команд следующий. Число 7, следующее непосредственно после `\parshape` и знака равенства, задает количество строк, имеющих нестандартные длину и/или отступ от левого поля. После этого числа, через пробел (конец строки, как мы помним, тоже пробел), перечислены отступы от левого поля и длины строк: 0cm — отступ первой строки от левого поля, 4cm — ее длина, 0.5cm — отступ второй строки от левого поля, 5cm — ее длина, и т. д. Если написано, что `\parshape` равно n , то после этого должно следовать $2n$ длин. Если реально в абзаце получится менее n строк, то указания на длину и отступ отсутствующих строк

будут проигнорированы $\text{\TeX}'ом$; если же строк, как в нашем примере, получается больше, чем n , то все последующие строки будут иметь те же отступ и длину, что заданы для строки номер n . Заметим, наконец, что абзац мы начали командой \noindent , чтобы отступ самой первой строки был действительно равен нулю (если абзац начинается не командой \noindent , а обычным образом, то в первой строке будет еще присутствовать пробел длиной в \parindent).

После пустой строки или команды \rag действие параметров, заданных командой \parshape , прекращается.

У абзаца, форма которого задана с помощью \hangindent или \parshape , длина и отступ строки зависят, как вы могли заметить, от ее номера. Если такой абзац содержит выключную формулу, то \TeX считает, что эта формула занимает три строки, причем сама формула расположена в средней из этих трех (реально формула может, разумеется, занимать больше места).

10. Линейки

10.1. Линейки в простейшем виде

Один из часто встречающихся элементов полиграфического оформления — так называемые «линейки». Например, в книге, которую вы читаете, колонтитулы отделены линейкой от основной части страницы. В $\text{\TeX}'е$ линейкой (rule по-английски) называется любой черный прямоугольник. Для создания линеек в $\text{\LaTeX}'е$ используется команда $\text{\rule{ширина}{высота}}$. У этой команды два обязательных аргумента: первый задает ширину прямоугольника-линейки, второй — высоту (оба этих размера должны быть заданы в используемых $\text{\TeX}'ом$ единицах измерения — см. с. 25). Линейка, созданная командой $\text{\rule{ширина}{высота}}$, рассматривается $\text{\TeX}'ом$ так же, как буква.

Если необходимо, чтобы созданный командой $\text{\rule{ширина}{высота}}$ прямоугольник был сдвинут по вертикали относительно уровня строки, надо воспользоваться командой $\text{\rule{ширина}{высота}}$ с необязательным аргументом. Этот аргумент — расстояние, на которое надо сдвинуть линейку по вертикали — ставится перед обязательными; если расстояние положительное, то сдвиг идет вверх, если отрицательное, то вниз. Пример:

В этом месте, прямо посреди абзаца, будет линейка ■, а после нее продолжится обычный текст. Сравните также ■ и ■

В этом месте, прямо посреди абзаца, будет линейка $\text{\rule{.5em}{15pt}}$, а после нее продолжится обычный текст. Сравните также $\text{\rule{5pt}{5pt}}$ и $\text{\rule[-3pt]{5pt}{5pt}}$.

10.2. ТЕХ'овские команды для генерации линеек

ЛАTeX'овская команда `\rule` обладает рядом недостатков. Например, то обстоятельство, что создаваемые с ее помощью линейки воспринимаются ТЕХ'ом как буквы, усложняют такую операцию, как печать линейки, простирающейся во всю ширину страницы. Если между абзацами, т. е. в «вертикальном режиме», сказать что-нибудь наподобие

```
\rule{10cm}{1pt}
```

то линейка начнется не с левого края текста, а после абзацного отступа: ТЕХ решит, что с этой «буквы» начинается новый абзац. Кроме того, при печати линеек с помощью команды `\rule` необходимо заранее знать их длину и ширину, что не всегда удобно (например, если линейка должна идти во всю ширину текста, то надо точно знать, чему эта ширина равна, либо, по крайней мере, знать, как она обозначается в ТЕХ'e). Избавиться от этого неудобства можно с помощью ТЕХ'овских команд `\hrule` и `\vrule`. Команда `\hrule` употребляется в «вертикальном режиме» (между абзацами). Она создает линейку высотой 0.4pt и шириной, равной ширине колонки текста. Команда `\vrule` употребляется в «горизонтальном режиме» (внутри абзацев). Она создает линейку шириной 0.4pt, простирающуюся по высоте до максимальной высоты букв в содержащей ее строке (если в строке присутствуют буквы наподобие `у`, опускающиеся ниже уровня строки, то и линейка будет опускаться ниже уровня строки). Пример:

Весь этот текст будет заключен между двумя линейками. Внутри абзаца тоже будет | линейка.

Если буквы в строках выше, то и линейка | будет больше.

```
\hrule\smallskip
```

Весь этот текст будет заключен между двумя линейками.

Внутри абзаца тоже будет `\vrule{}` линейка.

`\Large` Если буквы в строках выше, то и линейка `\vrule{}` будет больше. `\smallskip\hrule\exb`

Если вас не устраивает, что генерируемая командой `\hrule` линейка имеет высоту 0.4pt, то требуемую вам высоту можно указать в явном виде. Например, для задания линейки шириной во всю колонку и высотой 2 пункта надо написать (как водится, между абзацами) так:

```
\hrule height 2pt
```

Отсутствие символа `\` перед `height` не является опечаткой (`height` — не команда, а одно из так называемых «ключевых слов» TeX'a, наподобие уже встретившихся нам в разд. 8.4 слов `plus` и `minus`). Для явного задания ширины линейки, генерируемой командой `\vrule`, используется ключевое слово `width`:

```
\vrule width 2mm
```

В принципе можно указывать при команде `\hrule` не только высоту, но и ширину, а при команде `\vrule` — не только ширину, но и высоту, но в таком случае обычно проще воспользоваться И^AT_EX'овской командой `\rule`.

Если после команды `\hrule` или `\vrule` в тексте идет слово, совпадающее с одним из используемых этими командами ключевых слов (то бишь `height`, `width` или `depth`, о котором у нас речи не было), то это слово будет воспринято TeX'ом как ключевое, что приведет к сообщению об ошибке. При верстке текста на русском языке вероятность такого стечения обстоятельств исчезающе мала, но если вы хотите, чтоб неприятностей не было с гарантией, то после чего-нибудь вроде `\hrule height 2mm` пропустите строку (между абзацами это ничего не испортит), а после команды наподобие `\vrule width 2mm` поставьте еще команду `\relax`, означающую «ничего не делать».

10.3. Невидимые линейки

Высота и/или ширина линейки может быть и нулевой. Линейки нулевой высоты или ширины не печатаются, но тем не менее могут оказывать влияние на вид текста. Например, линейка нулевой ширины и ненулевой высоты занимает место по вертикали; если ее высота больше высоты букв в строке, то высота строки, содержащей эту невидимую линейку, увеличится:

Для того, чтобы раздвинуть две строки, можно в одну из них поместить невидимую линейку нулевой ширины и ненулевой высоты.

Для того, чтобы раздвинуть две строки, можно в одну из них поместить невидимую линейку `\rule{Opt}{5mm}` нулевой ширины и ненулевой высоты.

Один частный случай линейки нулевой ширины настолько важен, что в TeX'e и И^AT_EX'e для такой линейки предусмотрена специальная команда `\strut`. Невидимая линейка, создаваемая этой командой, имеет нулевую ширину; высота же ее установлена автором И^AT_EX'a с таким расчетом, чтобы она была чуть выше максимальной высоты букв текущего шрифта и опускалась ниже уровня строки настолько, насколько

могут опускаться буквы текущего шрифта. Например, в прямом светодиодном шрифте кегля 11 команда `\strut` создает линейку ширины 0, поднимающуюся над уровнем строки на 9.52 pt и опускающуюся ниже уровня строки на 4.08 pt.

Линейки нулевой ширины и ненулевой высоты действуют подобно команде `\vspace*`. Смысл невидимых линеек в том, что они позволяют создать вертикальные или горизонтальные пробелы в таких ситуациях, когда `\vspace` или `\hspace` не помогают. Вот пример ситуации, когда возникает нужда в невидимых линейках. Пусть в нашем тексте мы подчеркнули три слова подряд. Выглядит это не очень удачно: в словах с буквами вроде р, опускающимися ниже строки, линейки, подчеркивающие слово, также опускаются ниже строки, а хотелось бы, чтобы все эти линейки были на одном уровне. Выход из положения такой: добавить ко всем словам по невидимой букве, которая не занимает места по горизонтали, а по вертикали опускается на максимально возможное в текущем шрифте расстояние. В качестве такой буквы как раз и возьмем невидимую линейку, генерируемую командой `\strut`:

три слова подряд

```
\underline{три\strut}
\underline{\strut слова}
\underline{подряд\strut}
```

Как видите, `\strut` можно ставить хоть после слова, хоть перед ним (и даже посередине, если вы не запутаетесь с пробелами). Назначение этой команды в данном случае сводится к тому, чтобы не позволить линейке, подчеркивающей слово, подойти к этому слову слишком близко. Кстати, в переводе с английского слово `strut` означает «распорка».

В гл. II рассказывается про команду `\mathstrut`, выполняющую аналогичные функции в математических формулах.

Другие примеры использования невидимых линеек читатель найдет в главе, посвященной верстке с выравниванием; в главе о блоках мы также встретимся с линейками.

Глава IV

Оформление текста в целом

В этой главе мы рассмотрим такие вопросы, как общий стиль оформления документа, разбиение текста на разделы, титульный лист, оглавление и пр. *L^AT_EX* освобождает вас от многих забот об оформлении документа, но при этом и навязывает такие черты оформления, которые могут вас по тем или иным причинам не устраивать. От этого «диктата» можно отчасти отойти, если модифицировать стандартные стили (в последней главе мы расскажем, как это делать). В принципе можно создать и свой собственный стиль, весьма далекий от стандартных, но для этого требуются более глубокие познания в *TeX*'е, чем дает эта книга. Начнем же мы с того, что систематически рассмотрим вопрос о стандартных стилях.

1. Стили и стилевые опции в *L^AT_EX*'е 2.09

Команда `\documentstyle`, с которой начинается любой *L^AT_EX*'овский файл, имеет один обязательный аргумент — название основного стиля — и один необязательный, размещющийся перед обязательным — список «стилевых опций» (см. с. 19). Основной аргумент — это название «основного стиля» оформления документа.

В первой главе (см. разд. I.2.4) мы уже перечислили четыре основных стиля, предоставляемых *L^AT_EX*'ом: `article`, `report`, `book` и `letter`. Сейчас мы рассмотрим подробнее три основных стандартных стиля, предоставляемых нам *L^AT_EX*'ом. Стиль `letter` рассматривается ниже, в разд. 3.

Стиль `article` удобно применять для статей, стиль `report` — для более крупных статей, разбитых на главы, или небольших книг, стиль `book` — для книг. В табл. IV.1 перечислены некоторые черты оформления, присущие стандартным стилям. В ней знак «+» означает «всегда

Таблица IV.1

	article	report	book
Автоматически нумерующиеся разделы	+	+	+
Разбиение на главы	-	+	+
Разбиение на части	+	+	+
«Двусторонняя»* печать	⊤	⊤	+ (\pm в LATEX'e 2ε)
Титульный лист	⊤	+	+
			\pm в LATEX'e 2ε
Колонтитулы	⊤	\pm	\pm
Высота всех страниц одинакова	⊤	⊤	\pm
Набор в две колонки	⊤	⊤	⊤

* с разными полями для четных и нечетных страниц.

присутствует», знак «-» означает «всегда отсутствует», знак «⊤» означает «по умолчанию отсутствует, но будет присутствовать, если задать стилевую опцию или специальную команду», знак « \pm » означает «по умолчанию присутствует, но можно отменить с помощью стилевой опции или специальной команды». Мы не стремились охватить в этой таблице все детали отличий между стандартными стилями (например, колонтитулы в трех стилях оформляются немного по-разному).

Опишем теперь стилевые опции. Напомним (см. с. 19), что список стилевых опций через запятую ставится в квадратных скобках перед основным аргументом команды `\documentstyle`. Самые важные и часто употребляемые стилевые опции — это `11pt` и `12pt`. Они означают, что основной текст документа будет набран шрифтом кегля 11 или 12 соответственно. Если этих опций не указывать, то будет шрифт кегля 10.

Стилевая опция `twoside` задает печать с разными полями на нечетных и четных страницах (как в книгах). Ее можно указывать только для стилей `article` и `report`; в стиле `book` такая печать получается автоматически, и указывать опцию `twoside` при этом основном стиле нельзя.

ε В LATEX'e 2ε дело обстоит иначе. См. разд. IV.2.

Стилевую опцию `twocolumn` можно задавать для любого из трех основных стилей. Она означает, что набор текста будет производиться в две колонки. Так как абзацы при этом будут получаться довольно узкие, разумно при пользовании этой опцией заодно увеличить па-

метр `\tolerance` (см. с. 113), иначе будет получаться много строк, выбивающихся за колонку.

Стилевая опция `titlepage` применима только в том случае, если основной стиль — `article`. Если она задана, то у документа будет напечатан титульный лист (в стилях `report` и `book` титульный лист, как было указано выше, делается всегда).

Опция `draft` пригодна для любого из трех стилей. Если она включена, то каждая выбивающаяся на поля строка (т. е. строка, о которой выдается сообщение «`Overfull \hbox`» — см. с. 103), помечается на полях «марашкой» ■. Это удобно при подготовке корректур (английское слово `draft` как раз и означает «набросок»).

Режим, при котором разные страницы могут иметь разную высоту, задается, как мы помним, командой `\raggedbottom` (см. разд. III.8.7). По умолчанию такой режим устанавливается стилями `article` и `report`, если только в качестве стилевой опции не указана «двусторонняя» печать (стилевая опция `twoside`). Во всех остальных случаях \LaTeX будет, по умолчанию, делать все страницы одинаковой высоты.

Следующие две стилевые опции, применимые к любому из основных стилей, влияют на оформление выключных математических формул; если вы пропустили при чтении соответствующие места в гл. II, то пропустите и это место. Опция `fleqn` означает, что выключные формулы, заданные с помощью окружений `equation`, `eqnarray` и `displaymath`, а также пары команд `\[` и `\]`, будут напечатаны не в центре строки, а в ее левой части. Опция `leqno` означает, что номера формул, генерируемые окружениями `equation` и `eqnarray`, будут печататься не справа, а слева.

Титльному листу и разбиению документа на разделы будут посвящены отдельные разделы; чтобы завершить наш обзор различных стандартных вариантов стиля, нам осталось обсудить колонтитулы и номера страниц, чему будет посвящен разд. IV.4, а также дополнительные возможности, предоставляемые $\text{\LaTeX}'e$ 2 ε . Этому посвящен разд. IV.2.

2. Классы документов и их опции в $\text{\LaTeX}'e$ 2 ε

Как мы уже отмечали в первой главе, в $\text{\LaTeX}'e$ 2 ε файл должен начинаться с команды `\documentclass`, а то, что в $\text{\LaTeX}'e$ 2.09 называлось основным стилем, в $\text{\LaTeX}'e$ 2 ε называется классом документов. Кроме того, напомним, что после `\documentclass` в $\text{\LaTeX}'e$ 2 ε может идти одна или несколько команд `\usepackage`; аргумент этой команды — это список, через запятую, стилевых пакетов, подключаемых к нашему документу. В первой главе мы умолчали о том, что некоторые стилем-

вые пакеты допускают задание своих личных стилевых опций (каких именно — зависит от пакета). Список стилевых опций пакета задается в необязательном аргументе команды `\usepackage` (через запятую, если опций несколько). Необязательный аргумент команды `\usepackage` ставится перед обязательным.

Например, если включить в преамбулу строку

```
\usepackage{amsmath}
```

то вам откроются *AMS- \TeX* 'овские дополнительные возможности набора математических формул (см. приложение Е); если же написать

```
\usepackage[noamsfonts]{amsmath}
```

то у вас будут все эти возможности, кроме использования готического и ажурного шрифтов (см. с. 57).

Все, сказанное выше о стиле оформления документа, применимо и к $\text{\TeX}'у 2\varepsilon$ (надо только говорить «класс документа» вместо «основной стиль» и писать `\documentclass` вместо `\documentstyle`). Наряду с этим $\text{\TeX} 2\varepsilon$, как водится, предоставляет дополнительные возможности. Кроме трех основных классов `article`, `report` и `book`, в основной набор $\text{\TeX}'а 2\varepsilon$ входит класс `proc`; существуют также относящиеся к *AMS- \TeX* 'у классы документов `amsart`, `amsproc` и `amsbook` (мы рассмотрим их в приложении Е). Наконец, у классов `article`, `report` и `book` появились новые стилевые опции. С них мы и начнем.

В $\text{\TeX}'е 2.09$ размеры текста и полей, устанавливавшиеся $\text{\TeX}'ом$ автоматически, зависели только от стиля (и, кстати, плохо подходили к российским стандартам). В $\text{\TeX}'е 2\varepsilon$ можно указать стилевую опцию, задающую формат используемой бумаги, после чего \TeX рассчитает размеры текста и полей так, чтобы они максимально соответствовали этому формату. Эти опции таковы:

`a4paper` 210×297 миллиметров — самый ходовой в нашей стране;

`a5paper` 148×210 миллиметров;

`b5paper` 176×250 миллиметров;

`legalpaper` 8.5×14 дюймов;

`executivepaper` 7.25×10.5 дюймов.

Если ни одна из этих опций не указана, $\text{\TeX} 2\varepsilon$ считает, что размер бумаги равен 8.5×11 дюймов.

Если предполагается расположить текст так, чтобы он шел параллельно широкому, а не узкому краю бумаги, то можно указать опцию `landscape`: в этом случае \TeX будет вычислять размеры текста и полей, считая, что ширина и высота листа бумаги поменялись ролями. Подчеркнем, что задание опции `landscape` само по себе текст на 90° не повернет: он будет сверстан $\text{\TeX}'ом$ исходя из соответствующих размеров, но дальше необходимо иметь принтер и/или `dvi`-драйвер, способные обеспечить печать текста в такой ориентации (опция `landscape` не способна превратить узкий принтер в широкий). По умолчанию считается, что строки параллельны узкому краю листа.

Наряду с опцией `titlepage`, в $\text{\LaTeX}'e$ 2 ε все три основных класса документов допускают опцию `notitlepage`. Если ее указать, то отдельного титульного листа печататься не будет (даже в документе класса `book`).

Наряду с опцией `twoside`, в $\text{\LaTeX}'e$ 2 ε все три основных класса документов допускают опцию `oneside`. Если ее указать, то поля на четных и нечетных страницах будут одинаковы.

У классов `report` и `book` имеются опции `openright` и `openany`. Если указана опция `openright`, то каждая глава начинается обязательно с нечетной страницы (если необходимо, то ради этого печатается дополнительная пустая страница; на развороте нечетная страница будет правой). Если указана опция `openany`, то новая глава может начинаться как с четной, так и с нечетной страницы, и лишних пустых страниц ради начала главы \TeX не делает. По умолчанию в классе `report` \TeX действует так, как если бы было установлено `openany`, а в классе `book` — как если бы было `openright` (так же обстоит дело и в $\text{\TeX}'e$ 2.09, но там нет легального способа повлиять на то, с каких страниц будут начинаться главы).

Нам осталось сказать про класс документов `rgos`. Он предназначен для печати статей, входящих в сборники докладов на конференциях. В этом классе печать всегда в две колонки, с уменьшенными полями. Опции `a5paper`, `b5paper`, `onecolumn` и `titlepage` в классе `rgos` использовать нельзя. Нельзя также пользоваться маргиналиями (разд. 9).

При пользовании классом `rgos` внизу каждой страницы будет напечатано слово `Page` («страница») и номер страницы. Если вы хотите, чтобы вместо `Page` печаталось что-то другое, скажем, «с.», то нужно в преамбуле переопределить команду `\pagename`, то есть написать:

```
\renewcommand{\pagename}{с.}
```

(см. с. 154 по поводу переопределения «стандартных надписей», делаемых $\text{\TeX}'ом$ по-английски; `\renewcommand` обсуждается в гл. VII).

3. Деловые письма на L^AT_EX'e

L^AT_EX имеет стиль (в L^AT_EX'e 2ε — класс документов) `letter`, специально предназначенный для оформления деловых писем в соответствии с принятыми в США стандартами. Пользоваться им надо так. Во-первых, в преамбуле документа надо задать адрес отправителя в аргументе команды `\address` (можно применять \\ для ручного разбиения адреса на строки) и подпись отправителя в аргументе команды `\signature`. Можно также задать дату в аргументе команды `\date`. Если этого не сделать, то автоматически будет проставлена дата трансляции L^AT_EX'овского файла (по-английски).

Бежуло `\begin{document}` и `\end{document}` идет собственно текст письма или писем (в одном L^AT_EX'овском файле можно поместить несколько писем от одного отправителя). Каждое письмо оформляется как окружение `letter`. Это окружение принимает один обязательный аргумент — адрес того, кому предназначено письмо (опять-таки адрес можно разбивать на строки командами \\). Внутри окружения `letter` располагается собственно текст письма. Начинаться он должен командой `\opening`, в аргументе которой записывается вступительное обращение, а завершаться — командой `\closing`, в аргументе которой записывается заключительная фраза перед подписью («Искренне Ваш», например). На рис. IV.1 приведен пример оформленного по всем правилам L^AT_EX'овского файла с письмом, а на рис. IV.2 — то, как будет выглядеть это письмо на печати. Если вы хотите написать в письме

```
\documentstyle[11pt]{letter}
\address{129820, Москва, 1-й Рижский пер., д.2\\
Издательство \glqq Мир\grqq, СурTUG}
\signature{Исполнительный директор\\ И. А. Маховая}
\date{30 августа 1995 г.}
\begin{document}
\begin{letter}{Всем, кому нужно набирать\\
тексты с формулами}
\opening{Дорогие друзья!}
Пользуйтесь системой \TeX.
\closing{С наилучшими пожеланиями,}
\end{letter}
\end{document}
```

Рис. IV.1. Пример делового письма

что-то после вашей подписи, следует после команды `\closing` дать ко-

129820, Москва, 1-й Рижский пер., д.2
Издательство „Мир“, СурТУГ

30 августа 1995 г.

Всем, кому нужно набирать
тексты с формулами

Дорогие друзья!

Пользуйтесь системой *TeX*.

С наилучшими пожеланиями,

Исполнительный директор
И. А. Маховая

Рис. IV.2. Письмо с рис. IV.1 в напечатанном виде

манду `\ps` («постскриптуm»), и уже после нее — сам текст (команда `\ps` необходима, но никакого текста, даже просто P.S., она не генерирует).

В преамбуле можно дать команду `makelabels`. В этом случае на отдельной странице будут напечатаны адреса для всех писем (окружений `letter`), входящих в файл.

Все описанные выше стилевые опции применимы и к документам стиля/класса `letter`.

После команды `\closing` можно дать команду `\cc`, в аргументе которой указывается, кому вы собираетесь отослать копии письма. Можно также дать команду `\encl`, в аргументе которой указывается список вложений (например, если ваше письмо — сопроводительное к пакету документов). Американский стандарт на оформление таких вещей несколько расходится с отечественным. Если, тем не менее, вы решитесь воспользоваться этими средствами в письме на русском языке, следует изменить значение команд `\ccname` и/или `\enclname` с помощью `\renewcommand` (см. образцы в разд. 6.3).

4. Стиль оформления страницы

Для задания стиля оформления страницы в \LaTeX 'е предусмотрена команда `\pagestyle`. Эта команда имеет один обязательный аргумент — слово, обозначающее этот стиль. При использовании стандартными стилями \LaTeX 'а это слово должно быть одним из следующих:

`empty` Нет ни колонтитулов, ни номеров страниц.

`plain` Номера страниц ставятся внизу в середине строки, колонтитулов нет.

`headings` Присутствуют колонтитулы (включающие в себя и номера страниц).

`myheadings` Присутствуют колонтитулы, оформленные так же, как в предыдущем случае; отличие в том, что текст, печатающийся в колонтитулах (в стандартном случае это номера и названия разделов документа), не порождается \LaTeX 'ом автоматически, а задается пользователем в явном виде.

Если основной стиль — `article`, то по умолчанию страницы оформляются стилем `plain`, в двух других основных стилях — стилем `headings`. «Стиль» `myheadings` мы рассмотрим в разд. IX.6.

Наряду с командой `\pagestyle`, задающей стиль оформления всех страниц, есть и команда `\thispagestyle`, задающая стиль оформления одной отдельно взятой страницы. Она принимает такой же аргумент, как и `\pagestyle`, но указываемое этим аргументом оформление относится только к той странице, на которую попал текст, окружающий эту команду. Заранее предугадать, на какую страницу попадет данный фрагмент текста, обычно невозможно. Поэтому, если хотите от этой команды предсказуемых результатов, употребляйте ее непосредственно после `\newpage` или `\clearpage`.

Можно при желании сделать так, чтобы страницы нумеровались не арабскими цифрами, что делается по умолчанию, а римскими цифрами или буквами в алфавитном порядке. Для этого предназначена команда `\pageumbering`. Она имеет один обязательный аргумент, который может быть одним из следующих:

<code>arabic</code>	арабские цифры (1, 2, 3 ...)
<code>roman</code>	римские цифры (i, ii, iii ...)
<code>Roman</code>	римские цифры (I, II, III ...)
<code>alph</code>	строчные буквы (a,b,c ...)
<code>Alph</code>	прописные буквы (A,B,C ...)

Команда `\pagenumbering` не только меняет вид, в котором на печати представляются номера страниц, но и начинает счет страниц заново (это удобно, например, в тех случаях, когда страницы предисловия надо нумеровать римскими цифрами, а страницы основного текста заново нумеровать арабскими). Поэтому разумно давать эту команду сразу же после `\newpage` или `\clearpage`.

На этом мы прерываем наше обсуждение того, как изменять стандартное оформление страницы. На самом деле можно изменить гораздо больше, переделав оформление колонтитулов или номеров страниц совершенно радикально. Речь об этом пойдет в гл. IX.

5. Поля, размер страницы и прочее

Стандартные \LaTeX 'овские стили самостоятельно устанавливают значения таких параметров, как ширина и высота страницы, размеры полей и пр. (в $\text{\LaTeX}'e 2e$ — с учетом стилевой опции, указывающей формат бумаги). Если эти значения вас не устраивают, их можно изменить. В настоящем разделе рассказано, как это сделать.

Размеры текста на странице, полей и пр. задаются различными параметрами со значением длины (см. разд. I.2.6 по поводу $\text{\TeX}'овских$ параметров). Прежде чем мы скажем, какими именно, надо предупредить читателя, что значения параметров, о которых ниже пойдет речь, можно менять только в преамбуле документа (или в стилевом файле, если вы создали собственный стилевой файл). Изменение этих параметров после `\begin{document}` в одних случаях вообще ничего не даст, а в других — приведет к нелепым результатам.

5.1. Ширина

Ширина текста на странице задается параметром `\textwidth`; если набор осуществляется в две колонки, то `\textwidth` включает в себя ширину обеих колонок и пробел между ними. Если вам нужно, чтобы ширина текста на странице равнялась 7 см, то напишите в преамбуле так:

```
\textwidth=7cm
```

При изменении ширины текста часто приходится менять и поля. В $\text{\LaTeX}'e$ предусмотрен параметр, регулирующий размер левого поля (коль скоро левое поле и `\textwidth` заданы, правое поле, как вы догадываетесь, определяется автоматически). Способ задания левого поля зависит от того, является ли набор в данном стиле «двусторонним» или нет. На с. 142 объяснялось, что при двустороннем наборе на страницах

с четными и нечетными номерами оставляются разные поля. В стилях (в *L^AT_EX'*е 2_{*c*} — классах документов) *article* и *report* набор по умолчанию односторонний, но он будет двусторонним, если указать стилевую опцию *twoside*. В стиле *book* набор является двусторонним всегда (в *L^AT_EX'*е 2_{*c*} можно сделать его односторонним и в классе *book*, если указать стилевую опцию *oneside*).

При одностороннем наборе величина левого поля задается параметром *\oddsidemargin*. При этом поле отсчитывается не от самого края листа: предварительно делается отступ в один дюйм. Таким образом, если вы скажете в преамбуле

```
\oddsidemargin=0pt
```

то текст будет начинаться на расстоянии один дюйм от края, а если будет сказано

```
\oddsidemargin=5mm
```

то отступ от края бумаги составит 30.4 мм (вспомним, что один дюйм равен 2.54 см). Если присвоить параметру *\oddsidemargin* отрицательное значение, то расстояние от края листа до начала текста будет, соответственно, меньше дюйма. Нелишне также напомнить, что когда вы присваиваете параметру со значением длины нулевое значение, то все равно должна быть указана какая-то единица длины (как у нас в примере); запись наподобие

```
\oddsidemargin=0
```

является ошибочной.

Все сказанное относилось к одностороннему набору. При двустороннем наборе параметр *\oddsidemargin* также используется, но смысл его несколько иной: на сей раз он задает размеры левого поля только для страниц с нечетными номерами. Что же касается страниц с четными номерами, то размеры левого поля для них задаются параметром *\evensidemargin*.

При наборе текста в две колонки используются еще два параметра. Во-первых, параметр *\columnsep* задает расстояние между колонками; во-вторых, колонки можно при желании разделить не только пробелом, но и вертикальной линейкой. Ширина этой линейки задается параметром *\columnseprule*. В стандартных стилях значение этого последнего параметра установлено равным нулю, так что линейка между колонками не печатается; чтобы линейка была, надо в преамбуле присвоить параметру *\columnseprule* значение, отличное от нуля (в этом случае ширина разделяющей колонки линейки включается в *\columnsep*).

5.2. Высота

Размер верхнего поля задается параметром `\topmargin`; как и в случае с левым полем, это — расстояние не непосредственно от края листа, а от линии, параллельной краю и отстоящей от него на один дюйм. При этом надо сознавать не только *от чего*, но и *до чего* отсчитывается это расстояние: именно, `\topmargin` — это расстояние до колонтитула. Если же колонтитул на странице отсутствует (например, потому, что он не предусмотрен стилем), то вверху страницы дополнительно будет пустое пространство, размер которого равен месту, отводимому на колонтитул (если вы доберетесь до гл. IX, то узнаете, что размер этого места задается параметром `\headheight`), плюс пустое пространство, равное отступу между колонтитулом и основным текстом (оно обозначается `\headsep`).

Высота текста задается параметром `\textheight`. При исчислении этого размера не учитываются ни номера страниц, ни колонтитулы, так что, если они предусмотрены стилем, полная высота текста на странице будет больше, чем `\textheight`.

Высоту страницы также можно изменять, присваивая в преамбуле параметру `\textheight` новое значение, но если стиль предусматривает, что все страницы должны иметь одинаковую высоту (см. с. 142 и ниже по поводу того, когда именно так бывает), то высоту текста нельзя устанавливать совсем уж произвольно: необходимо согласовать ее значение с параметрами `\topskip` и `\baselineskip`. Не вдаваясь в подробности, скажем, что первый из этих параметров определяет расстояние от низа первой строки¹⁵ до «верхнего обреза» основного текста страницы, в то время как параметр `\baselineskip` определяет расстояние между строками и зависит от используемого шрифта (будем надеяться, что вы не станете менять его значение, не изучив предварительно книги [3]). Так или иначе, значение `\textheight` следует устанавливать таким образом, чтобы отношение

$$\frac{\textheight - \topskip}{\baselineskip}$$

было целым числом. В L^AT_EX'овском стандарте `\topskip` всегда равен 10 пунктам. Что же до `\baselineskip`, то он равен 12 пунктам, если основной шрифт кегля 10, 13.6 пункта, если основной шрифт кегля 11, и 15 пунктам в кегле 12.

¹⁵Точнее говоря, от ее базисной линии: см. гл. VIII.

5.3. Сдвиг страницы как целого

Иногда при печати вы можете с удивлением обнаружить, что реальные расстояния от текста до края листа не такие, как предписано параметрами наподобие `\topmargin`. Дело в том, что у принтера, которым вы пользуетесь, могут быть свои представления о том, где находится край листа бумаги. Чтобы эти представления соответствовали реальности, понадобится настройка принтера и/или `dvi`-драйвера, используемого вами для распечатки. Если вам жаль тратить время на такую работу, можно просто изменить расположение всей страницы в целом на печатном листе. Для этого следует установить (в преамбуле) значения двух `TeX`'овских параметров: `\hoffset` и `\voffset`. Например, если в преамбуле написано

```
\hoffset=-5mm
\voffset=4.2mm
```

то вся страница в целом (со всеми колонтитулами, номерами страниц и пр.) будет сдвинута при печати на 5 мм влево и на 4,2 мм вниз.

6. Разделы документа

Работая с `ЛATeX'ом`, разумно делать заголовки и нумерацию разделов не вручную, а с помощью специальных команд. Сначала разберем, какими пользоваться, на примере команды `\section`.

6.1. Команда `\section`

Пусть вам нужно начать раздел документа, озаглавленный «Кое-что о слонах». Для этого в исходном тексте можно написать так:

```
\section{Кое-что о слонах}
```

Команда `\section` принимает один обязательный аргумент — название раздела (это же название пойдет в колонтитулы, если таковые предусмотрены стилем, и в оглавление, если вы дадите команду «создать оглавление» — см. ниже с. 158). Промежутки между разделами, их нумерация, те же колонтитулы — все это делается автоматически.

Кроме обязательного аргумента у команды `\section` предусмотрен и чеобязательный. Необязательный аргумент идет перед обязательным; в нем записывается вариант заголовка, предназначенный для оглавления и колонтитулов (если стиль предусматривает, что заголовок войдет в колонтитул). Обычно этот вариант — просто сокращенный заголовок. Пример:

\section[0 слонах]{Кое-что о слонах}

Необходимость в сокращенном варианте заголовка возникает, когда оказывается, что заголовок по длине не помещается в колонтитул. Это, конечно, будет видно при просмотре; кроме того, при трансляции в этом случае выдается такое сообщение:

```
Overfull \hbox has occurred while \output was active.
```

Раздел можно пометить командой `\label` (см. с. 26). После этого команда `\ref` будет выдавать номер этого раздела. Пример:

3.2 Кое-что о слонах

В этом разделе нашей книги речь пойдет в основном о слонах. Слоны (см. определение в разд. 3.2) — большие и добрые животные.

```
\section{Кое-что о слонах}
\label{elephants}
В этом разделе нашей книги
речь пойдет в основном о
слонах. Слоны
(см. определение в
разд. "\ref{elephants}") ---  
большие и добрые животные.
```

Как обычно с командами автоматической генерации ссылок, при первом запуске \LaTeX 'а будет выдано сообщение о том, что метка неизвестна, а в дальнейшем, если номер помеченного раздела изменится, \LaTeX выдаст предупреждение о том, что надо запустить его еще раз.

Иногда хочется, чтобы сокращенный вариант заголовка попал только в колонтитул, а в оглавлении был его полный вариант. Как этого добиться, рассказано в разд. IX.6 (см. особенно с. 298 и далее).

У команды `\section` есть вариант «со звездочкой» (см. с. 25). Команда `\section*` начинает новый раздел, не нумеруя его; на оглавлении и колонтитулах наличие раздела, вводимого этой командой, никак не отразится. У команды `\section*` предусмотрен только обязательный аргумент.

6.2. Какие бывают разделы документа

Теперь перечислим все команды для задания разделов документа, предоставляемые стандартными стилями \LaTeX 'а. Большинство из них работают совершенно аналогично команде `\section`; все отличия мы сейчас перечислим.

Для оформления разделов существуют такие команды:

```
\part \chapter \section \subsection
\subsubsection \paragraph \ subparagraph
```

В этом перечне каждая последующая команда обозначает более мелкий подраздел, чем предыдущая. Следует иметь ввиду, что команда `\chapter` («глава») в стиле `article` не определена (благодаря этому обстоятельству статью легко переделать в главу книги), остальные команды определены во всех трех стандартных стилях.

Стандартные стили обеспечивают нумерацию разделов, при которой более мелкий раздел «подчинен» более крупному: когда, например, начинается новый раздел `\section`, нумерация разделов `\subsection` и более мелких начинается заново. Исключением из этого правила является команда `\part` («часть»): если часть 2 кончается главой 5, то первая из глав части 3 будет иметь номер 6, а не 1. При модификации стандартных стилей можно менять как принцип нумерации разделов, так и вид этих «номеров» на печати (например, если мы захотим, чтобы разделы обозначались последовательными буквами алфавита).

Все то, что мы говорили про необязательный аргумент и вариант «с звездочкой» у команды `\section`, применимо и к командам, перечисленным в этом разделе. «Слишком мелкие» разделы, согласно стандартным стилям, не отражаются ни в оглавлении, ни в колонтитулах и не нумеруются, но, если вы употребите задающие их команды с необязательным аргументом или со звездочкой, ошибкой это не будет.

В разделах, создаваемых описанными выше командами, первый абзац набирается без абзацного отступа (за исключением самого мелкого раздела `\subparagraph`), причем L^AT_EX устроен таким образом, что создать этот отступ вам так просто не удастся. Если вы хотите, чтобы отступ в первом абзаце все-таки присутствовал, обратитесь к гл. IX, посвященной модификации стандартных стилей.

• Все сказанное выше применимо и к L^AT_EX'у 2ε, если заменить слово «стиль» на «класс».

6.3. Изменение стандартных заголовков

Возможно, вы уже обратили внимание, что главы, создаваемые L^AT_EX'ом с помощью команды `\chapter`, так и называются «Chapter», а не «Глава». Это — один из нескольких случаев, когда стили L^AT_EX'a используют для различных стандартных надписей английские слова. Если вместе с L^AT_EX'ом вы получили русифицирующий стилевой файл, то у вас есть возможность указать стилевую опцию, делающую эти надписи русскими. Если такой русификации у вас нет, то придется решить эту проблему своими силами. Для этого в преамбулу документа надо внести следующую команду:

```
\renewcommand{\chaptername}{Глава}
```

Делать это надо, конечно, только в том случае, если команда `\chapter` предусмотрена в используемом вами стиле (для $\text{\LaTeX}'\text{e } 2\varepsilon$ — классе). Команда `\renewcommand`, используемая для переопределения значений уже существующих команд, объясняется в гл. VII.

Приведем заодно список остальных надписей, делаемых $\text{\LaTeX}'\text{om}$ по-английски, вместе с их русскими переводами и командами, которые надо переопределить с помощью `\renewcommand` для того, чтобы на печати появлялись именно эти переводы. В нескольких ближайших разделах мы объясним, как именно получить на печати упоминаемые в этой таблице список таблиц, список иллюстраций и т. п.

<code>\chaptername</code>	Chapter	Глава
<code>\contentsname</code>	Contents	Оглавление (или Содержание)
<code>\listfigurename</code>	List of Figures	Список иллюстраций
<code>\listtablename</code>	List of Tables	Список таблиц
<code>\abstractname</code>	Abstract	Аннотация
<code>\refname</code>	References	Список литературы
<code>\bibname</code>	Bibliography	Список литературы
<code>\indexname</code>	Index	Предметный указатель
<code>\figurename</code>	Figure	Рис.
<code>\tablename</code>	Table	Таблица
<code>\partname</code>	Part	Часть
<code>\appendixname</code>	Appendix	Приложение

Пояснений эта таблица, видимо, не требует, за одним исключением: если стиль (в $\text{\LaTeX}'\text{e } 2\varepsilon$ — класс) документа есть `article` (или `proc` — это опять для тех, кто уже перешел на $\text{\LaTeX} 2\varepsilon$), то для получения русского названия списка литературы надо переименовать команду `\refname`, а если `report` или `book`, то надо переименовать команду `\bibname`. Пожалуйста, не перепутайте эти два случая, иначе \LaTeX зафиксирует ошибку и выполнять вашу команду откажется.

6.4. До и после основного текста: аннотация, приложение и т. п.

В стилях (классах) `article` и `report` (в $\text{\LaTeX}'\text{e } 2\varepsilon$ и в классе `proc`) предусмотрена возможность оформить аннотацию ко всему документу. Это делается с помощью окружения `abstract`. До начала основного текста следует поместить текст аннотации, заключенный между `\begin{abstract}` и `\end{abstract}`. Этот текст будет автоматически озаглавлен «Abstract», если не менять стиля. Чтобы получить другой заголовок, переопределите команду `\abstractname` (см. предыдущий пункт).

Команда `\appendix` означает, что с этого места начинается приложение к документу. Сама она никакого текста не производит, а делает вот что:

- Начинает заново нумерацию разделов документа.
- «Самые крупные» разделы документа (`\section` в стиле (классе `article` и классе `proc`, `\chapter` в двух других основных стилях/классах) начинают нумероваться не цифрами, а прописными латинскими буквами.
- Если определена команда `\chapter`, то главы будут называться не «Chapter», а так, как определено в команде `\appendixname` (см. предыдущий пункт).
- ε Если вы пользуетесь $\text{\LaTeX}'\text{ом } 2_{\varepsilon}$ и класс вашего документа — `book`, то можете воспользоваться услугами команд `\frontmatter`, `\mainmatter` и `\backmatter`. Командой `\frontmatter` открывается «вводная часть» книги: после этой команды страницы начинают нумероваться римскими цифрами (как если бы мы сказали `\pagenumbering{roman}`). В той части текста, на которую распространяется действие команды `\frontmatter`, главы не нумеруются. Переход к основной части книги задается командой `\mainmatter`: печать начинается с новой страницы, страницы начинают нумероваться заново арабскими цифрами. Наконец, команда `\backmatter` задает переход к заключительной части книги: ε главы перестают нумероваться (нумерация страниц не меняется).

7. Титульный лист, оглавление, список литературы

7.1. Титульный лист

Для того чтобы оформить заголовок ко всему документу, надо сделать две вещи: задать информацию для заголовка (автор, название и т. п.) и дать $\text{\LaTeX}'\text{у}$ команду этот заголовок сгенерировать. Второе делается с помощью команды `\maketitle`. Она создаст титульный лист, если это предусмотрено стилем (стало быть, для стилей `report` и `book` — всегда, для стиля `article` — если указана стилевая опция `titlepage`).

- ε В $\text{\LaTeX}'\text{е } 2_{\varepsilon}$ вместо «если это предусмотрено стилем» следует сказать «если это предусмотрено классом документа и/или стилевыми опциями». См. с. 145.

Если титульный лист не предусмотрен, то команда `\maketitle` разместит заданную вами информацию об авторе, заглавии и прочем на первой странице (выбрав подходящие шрифты и сделав подобающие отступы между титульной информацией и текстом).

Так как команда `\maketitle` генерирует текст, ее нельзя помещать в преамбуле документа.

Теперь объясним, как задавать \LaTeX у информацию для титула. Автор задается с помощью команды `\author`. Она принимает единственный обязательный аргумент — имя автора (в том виде, как вы хотите его видеть на титуле). Если авторов несколько, их имена должны быть разделены командой `\and`.

Заглавие задается с помощью команды `\title`. Если заглавие длинное, можно самому задать его разбиение на строки с помощью команды `\\\` (кстати, в этой ситуации защищать ее с помощью `\protect` не требуется — см. с. 164); если этого не сделать, заглавие будет разбито на центрированные строки автоматически, как если бы это был абзац в окружении `center` (см. с. 126, а также пример на с. 24).

Следующий элемент информации для титула — команда `\date`. Она имеет один обязательный аргумент, в котором можно задать любой текст (например, дату, в согласии с переводом слова `date`), который будет размещен на титульном листе (или перед началом основного текста, если титульный лист не предусмотрен стилем и/или опциями) в одной или нескольких центрированных строках (так же, как и текст, задаваемый в аргументе команды `\title`). В частности, можно оставить аргумент этой команды «пустым», если сказать `\date{}` — тогда соответствующий текст вообще не появится. Но если вы вообще не дадите эту команду, хотя бы и с пустым аргументом, то \LaTeX напечатает на титуле дату своего запуска, причем по-английски.

Команды `\author`, `\title` и `\date` можно давать в любом порядке, но обязательно до команды `\maketitle` (можно и в преамбуле). Команда `\maketitle` должна быть первой из команд, генерирующих текст.

Наконец, последнее, что можно сделать с информацией для титула документа, — это сделать сноски. К любому из авторов, к любым словам в титуле или в тексте, содержащемся в аргументе команды `\date`, можно сделать сноску с помощью команды `\thanks`, имеющей один обязательный аргумент — текст сноски (в отличие от обычных сносок в тексте, он должен состоять из одного абзаца).

Сноски будут напечатаны внизу титульного листа (или первой страницы, если титульный лист не предусмотрен). Пример задания сносков:

```
\author{Борис Заходер}
\title{Винни-Пух и все-все-все}
\thanks{Вообще-то}
```

```
это перевод из А."А."Милна})  
\date{}
```

Обратите внимание, что команда `\thanks` помещается *внутри* аргумента команд `\title` и/или `\author`.

Наконец, можно при желании вообще не использовать стиль оформления титульного листа, диктуемый нам \LaTeX'om . Сделать это очень просто — надо воспользоваться окружением `titlepage`. Текст между `\begin{titlepage}` и `\end{titlepage}` составит титульный лист, за оформление которого целиком отвечает тот, кто текст готовит. \LaTeX внутри этого окружения делает только три вещи:

- Устанавливает печать в одну колонку (даже если сам документ будет печататься в две колонки).
- Начинает новую страницу и устанавливает счетчик числа страниц в нуль.
- Устанавливает странице стиль оформления `empty` (без колонтитула и номера).

Что и как разместить на этой странице — ваша забота.

7.2. Оглавление

В процессе работы \LaTeX автоматически собирает информацию для создания оглавления и записывает ее в специальный файл с тем же именем, что у обрабатываемого файла, и расширением `.toc`. Чтобы \LaTeX записал эту информацию, а затем воспользовался ею и напечатал оглавление, надо дать команду `\tableofcontents`.

Стало быть, оглавление, генерируемое \LaTeX'om , всякий раз будет «на шаг отставать» от реального положения дел. Чтобы учесть все возможные изменения и получить верное оглавление, надо будет в самом конце работы над текстом запустить \LaTeX еще раз (напоминания об этом \LaTeX не выдаст).

Все оглавление в целом будет озаглавлено словом, определяемым командой `\contentsname` (см. разд. 6.3).

Если вас не устраивает стандартный стиль оформления оглавления, прочтите в последней главе, как его можно изменить.

7.3. Список литературы

\LaTeX предоставляет возможность оформить список литературы, элементы которого нумеруются автоматически; в тексте при этом надо

ссылаются не на эти номера, которые могут измениться в процессе работы над документом, а на установленные вами условные обозначения для элементов списка литературы (будем для краткости называть их «источниками»).

Список литературы оформляется как окружение `thebibliography`. Это окружение имеет обязательный аргумент — номер элемента библиографии, который займет больше всего места на печати (в стандартных шрифтах все цифры имеют одинаковую ширину, так что достаточно привести в качестве аргумента, например, номер 99, если в списке литературы будет заведомо меньше 100 источников).

Каждый источник вводится командой `\bibitem`. У нее есть один обязательный аргумент — ваше условное обозначение. В качестве такого обозначения можно использовать любую последовательность из букв и цифр.

В тексте ссылка на источник делается с помощью команды `\cite`. У нее есть обязательный аргумент — условное обозначение того источника, на который вы ссылаетесь. Можно сослаться сразу на несколько источников — для этого в аргументе команды `\cite` надо указать через запятую обозначения для тех источников, на которые вы хотите сослаться. Приведем пример (в котором для экономии места мы опустили заголовок «Список литературы»):

В книге [3, Глава 1] описана встреча Винни-Пуха с несколькими пчелами. В [2, 1] приведены другие сведения о медведях.

- [1] М. Е. Салтыков-Щедрин.
Медведь на воеводстве.
- [2] Л. Н. Толстой. Три медведя.
- [3] А. А. Милн. Винни-Пух.

```
В книге \cite[Глава 1]{Winnie}
описана встреча Винни-Пуха
с несколькими пчелами.
\begin{thebibliography}{99}
\bibitem{voevoda}
приведены другие
сведения о медведях.
\begin{thebibliography}{99}
\bibitem{med3} М.~Е.~Салтыков-Щедрин.
Медведь на воеводстве.
\bibitem{med3} Л.~Н.~Толстой.
Три медведя.
\bibitem{Winnie} А.~А.~Милн. Винни-Пух.
\end{thebibliography}
```

В этом примере вы также можете видеть команду `\cite` с необязательным аргументом: он ставится перед обязательным; в квадратных скобках записывается текст, который будет через запятую напечатан после номеров ссылок.

Как это обычно и происходит с автоматически генерируемыми L^AT_EX'ом ссылками, после первого запуска программы вы увидите сообщение о том, что ссылки не определены. Если в дальнейшем в процессе работы над текстом нумерация ссылок изменится, L^AT_EX сообщит вам об этом и предложит запустить программу еще раз, чтобы получить корректные ссылки.

Если вам не нравится, что источники в списке литературы нумеруются, можно придумать для них свои обозначения, которые будут печататься вместо номеров. Для этого надо использовать команду \bibitem с необязательным аргументом, идущим перед обязательным. В квадратных скобках ставится то обозначение, которое будет заменять номер для этого источника. Например, можно написать так:

```
\begin{thebibliography}{XXXX}
...
\bibitem[EGA]{Groth} A.~Grothendieck, J.~Dieudonné.
'Eléments de Géométrie Algébrique.
...
\end{thebibliography}
```

После этого команда \cite{Groth} будет генерировать текст [EGA].

Списку литературы в целом L^AT_EX автоматически дает заглавие, определяемое командой \refname в стиле (классе) article (в L^AT_EX'e 2_ε и в классе (proc)) и \bibname в стилях (классах) report и book (см. разд. 6.3). Если вас не устраивает, что это название — английское, его можно переопределить (см. там же).

7.4. Предметный указатель

L^AT_EX окажет вам помощь в создании предметного указателя к вашему документу. В отличие, однако, от списка литературы, который при использовании описанных выше команд \cite и \bibitem получается совершенно автоматически, процесс создания указателя автоматизирован в L^AT_EX'e не полностью. Именно, вы можете сделать две вещи:

- Если вам уже известно, на каких страницах расположены те термины, на которые вы собираетесь сослаться в указателе, вы можете организовать печать предметного указателя с помощью окружения theindex. Если предметный указатель должен завершать документ, то можно, на худой конец, напечатать весь документ, кроме указателя, и вручную выписать требуемые номера страниц¹⁶.

¹⁶У вас может возникнуть искушение пометить все места, на которые будете ссылаться, с помощью команды \label, а в окружении theindex сослаться на них

- Если первый способ вас не устраивает, то можно специальным образом пометить в файле термины, на которые вы собираетесь ссылаться в предметном указателе. При этом средствами $\text{\LaTeX}'\text{а}$ создается полуфабрикат, из которого предметный указатель получится после некоторой дополнительной обработки.

Разберем эти две возможности последовательно.

Внутри окружения `theindex` каждый элемент указателя вводится командой `\item`; команды `\subitem` и `\subsubitem` вводят элементы указателя, печатающиеся с дополнительными отступами (обычно это уточнения к заглавному слову). Наконец, команда `\indexspace` создает дополнительный вертикальный пробел (его можно использовать для отделения различных разделов указателя друг от друга):

Компьютеры 25–42	<code>\begin{theindex}</code>
IBM-совместимые 28	<code>\item Компьютеры 25--42</code>
ремонт 35	<code>\subitem IBM-совместимые 28</code>
цены 30	<code>\subsubitem ремонт 35</code>
болгарские 26	<code>\subsubitem цены 30</code>
Принтеры 40	<code>\subitem болгарские 26</code>
Кошки 120	<code>\item Принтеры 40</code>
Собаки 140–156	<code>\indexspace</code>
	<code>\item Кошки 120</code>
	<code>\item Собаки 140--156</code>
	<code>\end{theindex}</code>

Предметный указатель, получаемый из окружения `theindex`, печатается $\text{\LaTeX}'\text{ом}$ в две колонки (даже тогда, когда сам документ печатается в одну колонку). Кроме того, \LaTeX автоматически дает указателю заглавие, определяемое командой `\indexname` (см. разд. 6.3). Если вас не устраивает, что это название — английское, его можно переопределить (см. там же).

Теперь рассмотрим вторую возможность — как с помощью $\text{\LaTeX}'\text{а}$ автоматически создать полуфабрикат предметного указателя. Для этого нужно сделать две вещи. Во-первых, в преамбулу документа необходимо включить команду `\makeindex`. Во-вторых, при условии, что это сделано, можно пометить те места в тексте, на которые вы хотите сослаться в предметном указателе, командой `\index`. У этой команды один обязательный аргумент — текст вашей пометки (в простейшем случае такая пометка — это ключевое слово будущего предметного указателя). \LaTeX запишет информацию о том, на какие страницы попали

с помощью `\pageref`. При этом, однако, есть опасность, что $\text{\TeX}'\text{у}$ не хватит памяти для обработки такого огромного числа ссылок.

ваши пометки, в специальный файл с тем же именем, что и у вашего файла, и расширением `idx` (мы будем называть его `idx`-файлом). Пусть, например, в исходном файле встречались такие фрагменты:

`Многие люди любят домашних кошек.\index{Кошки}`

....

`Хорошо также иметь собаку.\index{Собаки}`

....

`Мало кто рискует держать дома такую диковинную кошку,\index{Кошки} как тигр.`

Предположим, что первая ссылка на кошек попала на страницу 5, ссылка на собак попала на страницу 7, а вторая ссылка на кошек попала на страницу 9. Тогда в `idx`-файл запишется вот что:

```
\indexentry{Кошки}{5}
\indexentry{Собаки}{7}
\indexentry{Кошки}{9}
```

Полученный таким образом `idx`-файл — это и есть полуфабрикат указателя, созданный `LATEX`'ом. Использовать этот полуфабрикат, однако же, еще нельзя по следующим причинам:

- Ссылки в `idx`-файле расположены не по алфавиту (или каким-то другим разумным образом), а записаны «в порядке поступления».
- В `idx`-файле может присутствовать несколько строк с одним заглавным словом и ссылками на разные страницы; в готовом указателе естественно было бы такие ссылки объединить.
- Все строки `idx`-файла равноправны, в нем отсутствует иерархия ссылок вроде той, что получается при использовании команд наподобие `\subitem` в окружении `theindex`.
- Команда `\indexentry`, с которой начинается каждая строка `idx`-файла, не определена в `LATEX`'е (это сделано сознательно!).

Поэтому, получив `idx`-файл, вы должны его каким-либо образом обработать; идеально, если в результате получится файл с отсортированными (по алфавиту, скажем) терминами и с командами `\item`, `\subitem` и т. д., так что можно будет написать

```
\begin{theindex}
\input <имя_файла>
\end{theindex}
```

Если вы умеете программировать, вы сможете осуществить такую обработку программным путем самостоятельно¹⁷; кроме того, к реализациям ТЕХ'а нередко прилагаются программы для обработки idx-файлов (если такая программа не рассчитана на работу с русскими буквами, то есть опасность, что русские слова она будет сортировать неправильно). Чтобы установить иерархию терминов (\subitem и т. п.), вам, возможно, придется поработать вручную (как отмечал Дональд Кнут, составление предметного указателя — процесс творческий, и полностью передоверять его компьютеру не следует). В приложении Г рассказано об одной свободно распространяемой программе обработки idx-файлов (кстати, эта программа позволяет автоматизировать и составление указателя с иерархией ссылок).

Как быть, если программы для обработки idx-файлов у вас нет и раздобыть или самостоятельно написать ее вы не можете? Во-первых, отсортировать строки idx-файла можно и без специальной программы, средствами текстового редактора (в хороших редакторах такая возможность обычно предусмотрена). После этого остается проблема, что делать с командами \indexentry. Если ваших программистских умений не хватает на то, чтобы преобразовать каждое \indexentry в \item, то есть еще одна возможность: осуществить это преобразование средствами самого ИТЭХ'а. Именно, после того, как вы отсортируете строки idx-файла (и сохраните, для надежности, отсортированный файл под другим именем, скажем, myindex.tex), надо определить оставленную неопределенной команду \indexentry таким образом, чтобы она делала ту же работу, которую призван делать \item. Для этого надо написать в преамбуле следующее:

```
\newcommand{\indexentry}[2]{\item #1 #2}
```

После этого ТЕХ будет воспринимать каждую запись вида

```
\indexentry{Кошки}{5}
```

так же, как если бы вместо этого было написано

```
\item Кошки 5
```

и можно будет просто написать в конце документа

```
\begin{theindex}
\input myindex.tex
\end{theindex}
```

¹⁷ Предостережение для непрофессионалов: написать правильно работающую программу сортировки непросто!

Пока что воспринимайте этот рецепт чисто догматически; по прочтении гл. VII, в которой подробно рассмотрен процесс определения новых команд, вы поймете, почему все получается именно так.

В аргументе команды `\index` могут быть любые символы, и вообще текст в аргументе этой команды может быть неосмысленным или недопустимым с точки зрения TeX'a — в любом случае аргумент команды `\index` будет в неизменном виде переписан в idx-файл. Смысл тут в том, что в аргументе команды `\index` можно задавать вспомогательную информацию для программы обработки idx-файла (примеры тому вы найдете в приложении Г). Единственное ограничение — не должно быть «несбалансированных» фигурных скобок (даже если эти скобки входят в состав команд `\{` или `\}`).

Если в файле присутствуют команды `\index`, но в преамбуле нет команды `\makeindex`, то в idx-файл записываться ничего не будет.

Наконец, еще одна тонкость: команду `\index` нельзя использовать внутри необязательного аргумента таких команд, как `\caption` или `\section`, `\chapter` и т. п.

7.5. Перемещаемые аргументы и хрупкие команды

Если в аргументе команды `\section` (или любой другой L^AT_EX'овской команды для создания раздела) присутствует не только текст, но и TeX'овские команды, то при трансляции они могут иногда вызвать сообщение об ошибке. Чтобы этого избежать, команду надо «защитить»: поставить непосредственно перед ней команду `\protect`. Приведем пример, когда возникает нужда в этой команде.

Пусть вы хотите включить в заголовок раздела ссылку на какую-то страницу документа. Для этого вы, как обычно, помечаете то место, на которое хотите сослаться, с помощью команды `\label`, а в заголовок раздела включаете команду `\pageref` со ссылкой на помеченное место. Для того, однако же, чтобы L^AT_EX обработал ваш текст правильно, надо в аргументе команды `\section` написать не просто `\pageref`, но `\protect\pageref`. В итоге ваш исходный файл должен будет выглядеть примерно так (несущественные для нас части файла мы заменили точками):

```
\documentstyle{article}
\begin{document}
\tableofcontents
```

Это --- начало документа.

.....

Здесь написано что-то очень важное.\label{metka}

.....

\section{Возвращаясь к напечатанному}

```
на с. ^\protect\pageref{метка}
.....
\end{document}
```

Если убрать в этом файле `\protect`, то после двух запусков \LaTeX 'а для его обработки на экране появится загадочное сообщение об ошибке.

Такого рода ситуация может возникать, когда \TeX 'овская команда является частью текста, который будет записан в специальный файл и использован при следующем запуске \LaTeX 'а (в нашем случае заголовок раздела записывается в файл с расширением `toc` для последующего использования в оглавлении). Если аргументом команды (в нашем случае команды `\section`) является такой текст, то этот аргумент называется *перемещаемым аргументом*; команды, которые, будучи использованы внутри перемещаемого аргумента, могут вызвать ошибку, называются *хрупкими командами*.

Из тех \LaTeX 'овских команд, которые могут реально понадобиться внутри заголовка раздела, большинство хрупкими не являются; наряду с `\pageref` и `\ref`, хрупкой является также команда `\`{}`, которая может понадобиться для указания места разрыва строки в заголовке. Если вы сомневаетесь, хрупкая или нет какая-то конкретная команда, можете спокойно ставить перед ней `\protect` — ничего плохого от этого не произойдет.

Не являются хрупкими и не нуждаются в защите с помощью команды `\protect` команды для расстановки диакритических знаков (см. с. 92), смены текущего шрифта и установки промежутков вручную (см. с. 91).

- ε При использовании \LaTeX 'ом 2ε необходимость в команде `\protect` будет возникать у вас реже, чем в \LaTeX 'е 2.09. В частности, команда `\pageref` и `\`{}` в \LaTeX 'е 2ε хрупкими не являются.

8. Плавающие иллюстрации и таблицы

Если вы хотите разместить в своем тексте иллюстрацию (в простейшем случае — место для приклеивания картинки, но это может быть и «псевдорисунок», создаваемый с помощью \TeX 'овского окружения `picture`, и импортированный графический файл, если ваш `dvi`-драйвер предоставляет такую возможность), то можно создать для нее «пустое место, не пропадающее при разрыве страницы», с помощью команды `\vspace*` (см. разд. III.8.4). Если, однако, таких команд будет много, то будет трудно (или вообще невозможно) найти подходящие места для разрыва страниц. В этом случае удобнее воспользоваться окружением `figure`.

Стоящий между `\begin{figure}` и `\end{figure}` текст автоматически размещается \LaTeX 'ом в таком месте, где он укладывается целиком (не переходя со страницы на страницу); это может быть не на «своей» странице, а позже. В последнем случае говорят, что иллюстрация «всплыла» на следующей странице; именно поэтому окружение `figure` называют еще «плавающей иллюстрацией».

Команда `\caption` позволяет сделать подрисуночную подпись. Эта команда имеет один обязательный аргумент — текст подписи. На печати подпись состоит из слова, определенного командой `\figurename` («Figure», если не переопределять эту команду — см. разд. 6.3), порядкового номера иллюстрации, присвоенного ей \LaTeX 'ом, и подписи, указанной в аргументе команды. Команду `\caption` можно давать в любом месте между `\begin{figure}` и `\end{figure}`: в соответствующем месте появится на печати и сгенерированная ею подпись. Разумно поэтому ставить команду `\caption` либо в конце окружения `figure` (тогда подпись будет размещена под иллюстрацией), либо в его начале (подпись появится над иллюстрацией).

Если команду `\label` поместить внутри окружения `figure` после команды `\caption`, то команда `\ref` будет генерировать номер иллюстрации. Например, рис. IV.3 на с. 167 получился так:

```
Например, рис. ^\ref{void} на с. ^\pageref{void} ...
\begin{figure}
\vspace{4cm}
\caption{Белый квадрат
на белом фоне}\label{void}
\end{figure}
```

На подписи к рис. IV.3 после номера стоит точка. К сожалению, согласно \LaTeX 'овскому стандарту в этом месте ставится не точка, а двоеточие, что в русском тексте выглядит неудачно. Чтобы получалась именно точка, надо использовать подходящую «русифицирующую» стилевую опцию (в \LaTeX 'е 2.09) или стилевой пакет (в \LaTeX 'е 2_e).

У окружения `figure` предусмотрен необязательный аргумент, с помощью которого можно высказать \LaTeX 'у свои пожелания по поводу размещения иллюстрации в тексте. Именно, после `\begin{figure}` (без пробела) можно поместить в квадратных скобках одну или несколько из следующих четырех букв, имеющих такие значения:

- т Разместить иллюстрацию в верхней части страницы.
- в Разместить иллюстрацию в нижней части страницы.
- р Разместить иллюстрацию на отдельной странице, целиком состоящей из «плавающих» иллюстраций (или таблиц — см. ниже).

Рис. IV.3. Белый квадрат на белом фоне

h Разместить иллюстрацию прямо там, где она встретилась в исходном тексте, не перенося ее никуда.

Если в квадратных скобках стоит несколько букв, это значит, что вы согласны на любой из предусматриваемых этими буквами вариантов. Если окружение `figure` задано без необязательного аргумента, это равносильно записи

```
\begin{figure}[tbp]
```

При наборе текста в две колонки полезно использовать не только само окружение `figure`, но и его вариант «со звездочкой» (см. разд. I.2.9): если сказать

```
\begin{figure*}
```

то при наборе текста в одну колонку это не будет ничем отличаться от окружения `figure` без звездочки, а при наборе текста в две колонки создаст иллюстрацию шириной в целую страницу (без звездочки получилось бы шириной в одну колонку). Если окружение открывается командой `\begin{figure*}`, то и закрываться оно должно командой со звездочкой.

Если при наборе в две колонки задать окружение `figure` (без звездочки) с необязательным аргументом `r`, то для печати иллюстраций будет выделена не отдельная страница, но отдельная колонка.

Окружение `table` определяет «плавающие таблицы». Все свойства этого окружения дословно совпадают с соответствующими свойствами окружения `figure`, за одним-единственным исключением: подпись, генерируемая командой `\caption`, начинается со слова, определенного в команде `\tablename` (см. разд. 6.3), и переопределять, при необходимости, надо именно эту команду. Кстати, подпись к таблице принято делать не снизу, как к иллюстрации, а сверху. Окружение `table*` при

наборе текста в две колонки определяет таблицы шириной в целую страницу.

В документе можно, при желании, получить автоматически сгенерированные списки иллюстраций и/или таблиц. Для этого используются команды `\listoffigures` (для иллюстраций) и `\listoftables` (для таблиц). Их работа аналогична команде `\tableofcontents`, генерирующей оглавление (см. с. 158): материал для этих списков собирается в специальные файлы с расширениями `lof` (для иллюстраций) и `lot` (для таблиц); при каждом запуске \LaTeX 'а информация, записанная в этих таблицах, относится к предыдущему запуску, так что в самом конце может понадобиться запустить \LaTeX лишний раз; наконец, команда `\caption` может принимать необязательный аргумент — вариант подписи под иллюстрацией или таблицей, предназначенный для включения в список иллюстраций или таблиц соответственно. Этот необязательный аргумент записывается (в квадратных скобках, как обычно) перед обязательным.

Как окружение `figure` не рисует картинок, так и окружение `table` только размещает таблицу на страницах документа, но не создает ее текста. Как набирать таблицы в \LaTeX 'е, мы расскажем в гл. VI.

В разд. IX.7 мы расскажем о том, как можно модифицировать оформление плавающих иллюстраций и таблиц.

8.1. Нештатные ситуации с плавающими объектами в \LaTeX 'е 2.09 и в \LaTeX 'е 2 ε

Когда вы набираете исходный текст, заранее неясно, куда именно попадут плавающие иллюстрации (или таблицы; далее мы не будем всякий раз делать этой оговорки). Поэтому при просмотре и пробной печати возможны всяческие неожиданности.

Начнем с неприятности, подстерегающей вас при пользовании весьма привлекательным необязательным аргументом `h` («печатать прямо здесь!») у окружения `figure` или `table`. Если при этом, к несчастью, расположить иллюстрацию именно в указанном месте невозможно (потому что посередине иллюстрации должно быть место разрыва страницы), то при пользовании \LaTeX 'ом 2.09 произойдет небольшая катастрофа: злополучная иллюстрация (а также, что особенно печально, все последующие) не появится на печати до ближайшей команды `\clearpage` (до конца главы, если стиль предусматривает разбиение на главы, а если нет, то и до самого конца текста).

ε \LaTeX 2 ε обрабатывает эту нештатную ситуацию более культурно: если иллюстрация с необязательным аргументом `h` попадает

на место разрыва страницы, то $\text{\LaTeX} 2\varepsilon$ действует так, словно в необязательном аргументе стояло не h , а ht . В результате иллюстрация будет напечатана вверху текущей или следующей страницы, а сообщение о происшедшем инциденте будет выдано на экран ϵ и в log-файл.

Причина вышеописанной катастрофы в том, что \LaTeX при размещении иллюстраций или таблиц никогда не нарушает той последовательности, в которой они заданы в исходном тексте. Это весьма похвально, но в случае, если возникла проблема с печатью одного из плавающих объектов (например, потому что на текущей и ближайших страницах иллюстрации уже занимают много места — в своем обычном режиме \LaTeX этого не любит), то не печатаются и все последующие. Сам Лесли Лэмпорт сравнивал эту ситуацию с завалом, возникающим на реке при лесосплаве. Расчистить такой завал можно с помощью команды $\backslash clearpage$ (с. 119), но при этом, вероятно, появится укороченная страница. Другой способ борьбы с такими неприятностями — менять различные параметры, определяющие предпочтения $\text{\LaTeX}'a$ по размещению плавающих иллюстраций. О том, что именно тут можно менять, рассказано в разд. IX.7.

В $\text{\LaTeX}'e 2\varepsilon$ есть несколько дополнительных способов борьбы с привидами плавающих иллюстраций.

Во-первых, команда $\backslash suppressfloats$ запрещает печать любых плавающих иллюстраций (а также таблиц; мы не будем всякий раз это оговаривать) на той странице, на которую эта команда попала. Можно применить команду и с необязательным аргументом: если написать

`\suppressfloats[t]`

то вверху данной страницы иллюстрации размещаться не будут; если в качестве необязательного аргумента указать b , то иллюстрации заведомо не появятся внизу данной страницы.

Во-вторых, $\text{\LaTeX} 2\varepsilon$ предоставляет вам средство не затруднить, а, наоборот, облегчить размещение плавающих объектов. Именно, в необязательном аргументе окружения `figure` или `table` можно перед буквой t , b или h поставить восклицательный знак. В этом случае при размещении плавающего объекта \LaTeX не будет обращать внимание на то, не слишком ли много иллюстраций попало на одну страницу и не слишком ли большую ее долю они займут (типичные причины, по которым \LaTeX обычно перемещает плавающие иллюстрации вперед по тексту). У иллюстрации, начинающейся с команды

`\begin{figure} [!b]`

больше шансов быть напечатанной безотлагательно, чем в случае, если бы восклицательного знака не было.

Наконец, если подключить стилевой пакет `afterpage`, то появится возможность «расчистки завала» без побочного эффекта в виде выдачи укороченной страницы: если вместо `\clearpage` сказать

`\afterpage{\clearpage}`

то «застрявшие» плавающие объекты будут напечатаны после того, как кончится текущая страница. Этим средством нельзя пользоваться при печати в две колонки.

9. Заметки на полях (маргиналии)

LATEX дает возможность делать заметки на полях страницы. Для этого используется команда `\marginpar` с единственным обязательным аргументом — текстом заметки. Если в исходном тексте будет написано

**Маргиналии (фонарики) --- заголовки в виде
надписей** `\marginpar{!!!}` на наружных полях страниц.

то на печати вы увидите:

!!! Маргиналии (фонарики) — заголовки в виде надписей на наружных полях страниц.

Название `\marginpar` является сокращением английских слов, означающих «абзац на полях». Впрочем, текст заметки может состоять и из нескольких абзацев, разделяемых, как обычно, пустыми строками.

Если документ печатается в одну колонку и в «одностороннем» стиле (т. е. основной стиль — `article` или `report` и нет стилевой опции `twoside`), то заметки выводятся по умолчанию на правое поле. Если документ печатается в одну колонку, но в «двустороннем» стиле — то на внешнее поле (правое, если страница имеет нечетный номер, и левое в противном случае). Если документ печатается в две колонки, то заметка всегда выводится на ближайшее поле (ближайшее к той колонке, в которую попала заметка).

У команды `\marginpar` предусмотрен и необязательный аргумент. Он размещается перед обязательным; если эта команда использована с необязательным аргументом, то текст, выводящийся на поля, будет зависеть от того, на правое или на левое поле попадает заметка: на правое поле будет выведен текст, приведенный в обязательном аргументе, на левое — текст, приведенный в необязательном аргументе. Таким образом можно, например, вывести на поля стрелку, указывающую на текст:

\marginpar[\$\Longrightarrow\$]{\Longleftarrow}

(см. с. 45 по поводу команд, генерирующих стрелки в математических формулах).

\LaTeX старается поместить заметки на полях на том же уровне, что и текст, к которым они относятся, но, если этих заметок на каждой странице получается помногу (как, например, в поэмах С. Т. Кольриджа «Сказание о старом мореходе» или В. В. Маяковского «Про это»), то некоторые из них, во избежание наложений, будут сдвинуты вниз, а иногда даже перенесены на другую страницу (\LaTeX сообщит об этом прискорбном событии во время трансляции).

Если текст набирается в одну колонку, то можно сделать так, чтобы заметки появлялись не на тех полях, на которых они должны быть согласно вышеописанным правилам, а на противоположных. Для этого надо дать команду \reversemarginpar. Существует еще и команда \normalmarginpar, возвращающая правила размещения заметок в исходное состояние.

Можно также менять параметры оформления самих заметок на полях. Эти параметры таковы:

\marginparwidth : Ширина строки в заметках на полях.

\marginparsep : Расстояние между полем и заметками.

\marginparpush : Минимальное расстояние по вертикали между соседними заметками.

Значения этих параметров устанавливаются автоматически, в зависимости от используемого стиля. Вам может понадобиться их изменить, если вы меняете размер полей и/или ширину текста и при этом хотите пользоваться командой \marginpar.

В некоторых ситуациях команду \marginpar применять нельзя. Например, она не может появиться внутри аргумента команды \mbox или внутри окружения, предназначенного для верстки таблиц (см. гл. VI). Если вы все-таки попробуете сделать заметку на полях к такому «запрещенному» месту, то \LaTeX выдаст сообщение об ошибке.

Глава V

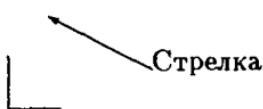
Псевдорисунки

Когда создавался \TeX , а начиналось это в 1978 году, компьютерная графика была развита еще слабо. Поэтому операция по включению в текст рисунков в виде графических файлов в $\text{\TeX}'$ е не стандартизована. Точнее говоря, \TeX допускает импорт графического файла в текст с помощью команды `\special`, в аргументе которой содержится информация об импортируемом файле, но способ задания этой информации не стандартизирован и зависит от конкретной реализации $\text{\TeX}'$ а (именно, от используемых *dvi*-драйверов), что снижает переносимость $\text{\TeX}'$ овских файлов¹⁸ (исходный текст, в котором `\special` не используется, обрабатывается совершенно одинаково на всех реализациях $\text{\TeX}'$ а на любом компьютере). Чтобы как-то сгладить это неудобство, создатель $\text{\LaTeX}'$ а Лесли Лэмпорт предусмотрел возможность создания примитивных рисунков, состоящих из прямых и наклонных (с ограниченным репертуаром наклонов) линий, стрелок и окружностей (в $\text{\LaTeX}'$ е 2 ε есть еще возможность рисовать кривые более или менее произвольной формы). В этой главе мы расскажем, как создавать такие «псевдорисунки».

1. Создание псевдорисунка и размещение в нем объектов

Псевдорисунки создаются с помощью окружения `picture`. Изучение этого окружения удобно начать с примера.

¹⁸ В приложении В мы расскажем о том, как это реализовано в популярном пакете *emTeX*



```
\begin{picture}(110,50)
\put(55,15){Стрелка}
\put(55,15){\vector(-2,1){40}}
\put(0,0){\line(1,0){20}}
\put(0,0){\line(0,1){20}}
\end{picture}
```

Разберем исходный текст, создавший этот «рисунок»: стрелку с надписью и уголок. На каждый псевдорисунок \LaTeX должен отвести в тексте определенное место (после чего сам рисунок вполне может и выйти за пределы отведенного места: все зависит от того, что и где вы будете «рисовать»). Эти размеры задаются в *круглых скобках* через запятую немедленно после `\begin{picture}`, сначала ширина, затем высота (команды, связанные с псевдорисунками — единственные в $\text{\LaTeX}'e$, у которых в определенных случаях обязательный аргумент ставится *не* в фигурных скобках). Между скобками, запятой и числами, задающими размеры псевдорисунка, не должно быть пробелов (помните, что конец строки также воспринимается $\text{\TeX}'om$ как пробел; если переноса на другую строку не избежать, воспользуйтесь знаком `%` для устранения получающегося пробела, как в примере на с. 17). По умолчанию ширина и высота псевдорисунка, и вообще все относящиеся к псевдорисункам размеры, задаются в пунктах (так и сделано в нашем примере). Можно указать любую единицу измерения размеров, относящихся к псевдорисункам: для этого надо изменить значение параметра `\unitlength` (см. с. 22 и далее по поводу параметров, являющихся длинами): если мы хотим, чтобы длины измерялись в миллиметрах, надо написать в преамбуле

`\unitlength=1mm`

(но не просто `mm!`). Размеры могут быть не только целыми, но и дробными числами, в которых нужно использовать десятичную точку (но не запятую!).

Итак, место на псевдорисунок выделено. Чтобы поместить что-то на этот псевдорисунок, используется команда `\put` (внутри окружения `picture` писать текст «просто так» не следует). После `\put` в *круглых скобках* через запятую следуют координаты того объекта, который мы размещаем на псевдорисунке (сначала абсцисса, затем ордината; началом координат по умолчанию считается левый нижний угол псевдорисунка), а затем, без пробела, в *фигурных скобках*, — тот объект, который надо нанести. Для первой из наших команд `\put` этот объект был просто текстом, и соответственно в *фигурных скобках* только этот текст и был; для остальных трех команд, размещавших на рисунке стрелку и два отрезка, в *фигурных скобках* помещается нечто более

сложное: описание этой стрелки и отрезков. В следующем разделе мы разберем, как такие описания устроены. Кстати, уголок в приведенном выше примере — не что иное, как левый нижний угол псевдорисунка (точка с координатами $(0,0)$).

Когда мы говорили о координатах объекта, имелись в виду координаты так называемой «точки отсчета» на этом объекте. Если объект — текст, то точка отсчета — его левый нижний угол. Иногда при размещении текста удобней задать координаты его правого, а не левого нижнего угла. Чтобы так сделать, можно воспользоваться командой `\llap` с одним аргументом — текстом, чья точка отсчета будет в правом нижнем углу. В следующем примере точка отсчета «полужирной» кошки будет в левом нижнем углу, а «рубленой» — в правом нижнем.

Кошка Кошка

```
\begin{picture}(110,40)
\put(52,20){{\bf Кошка}}
\put(50,20){\llap{\sf Кошка}}
\end{picture}
```

Точка отсчета стрелки — ее начало. Когда пойдет речь о других объектах, размещаемых на псевдорисунке, мы будем указывать, где расположены их точки отсчета.

Еще несколько общих правил, относящихся к окружению `picture`. Во-первых, внутри этого окружения не должно быть пустых строк. Во-вторых, необходимо сказать о том, как окружение `picture` взаимодействует с окружающим текстом. Весь псевдорисунок, порождаемый этим окружением, рассматривается \TeX ом как одна большая буква, ширина и высота которой заданы в скобках через запятую после `\begin{picture}`, так что если окружение `picture` встретилось в середине абзаца, эта «буква» будет помещена в строку, причем соседние строки раздвинутся, чтобы она поместилась. Если это не то, чего вы хотите, — начинайте окружение `picture` между абзацами (после пустой строки или команды `\par`). Можно также поместить окружение `picture` внутри окружения наподобие `flushright` или `center` — при этом \LaTeX автоматически установит разумные интервалы между псевдорисунком и окружающим текстом. Совершенно безбоязненно можно помещать окружение `picture` внутри «плавающего» окружения `figure` или `table` (см. разд. IV.8).

Кроме текста, на псевдорисунках можно размещать отрезки, стрелки, окружности, круги и овалы (прямоугольники с закругленными углами). Далее мы опишем, как задавать эти объекты.

2. Отрезки и стрелки

Отрезки задаются с помощью команды `\line`. И^AT_EX'у надо сообщить, каков наклон отрезка и каков его размер. Вот пример команды `\put`, выводящей отрезок:

```
\begin{picture}(100,50)
\put(60,50){\line(1,-2){20}}
\end{picture}
```

Как мы уже понимаем, здесь создается псевдорисунок размером 100 × 50 пунктов, на который наносится отрезок с началом в точке с координатами (60, 50). Наклон отрезка задается парой целых чисел, расположенных в *круглых скобках* через запятую непосредственно после `\line`. Отношение этих чисел должно быть равно «угловому коэффициенту» отрезка (тангенсу угла наклона к горизонтали); в нашем случае эти числа суть (1, -2), это означает, что отрезок отклоняется «на одну единицу вправо и на две единицы вниз». Если эти числа (1, 0), то отрезок горизонтален, если (0, 1), то отрезок вертикален.

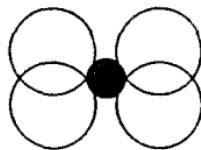
Размер отрезка задается в *фигурных скобках* после *круглых скобок*, в которых задан наклон. Этот размер, вообще говоря, — не его длина, но длина его проекции на горизонтальную ось (кроме случаев, когда отрезок вертикален — тогда задается его длина по вертикали).

Длину отрезка можно (если она не слишком мала) задавать произвольно, а вот наклон — нет. Каждое из целых чисел, задающих наклон, не должно превосходить 6 по абсолютной величине, и, кроме того, эти два числа не должны иметь общих делителей, кроме 1 (это последнее условие репертуара возможных наклонов не ограничивает).

Стрелки задаются с помощью команды `\vector`, которая нам уже встречалась в примере. Синтаксис этой команды совершенно такой же, как у команды `\line`: в *круглых скобках* пишется пара чисел, задающая наклон стрелки, а затем в *фигурных скобках* параметр, задающий ее размер (длина проекции на горизонтальную ось, если стрелка не вертикальна, и длина проекции на вертикальную ось, если стрелка вертикальна). Отличие от команды `\line` в том, что репертуар возможных наклонов стрелок еще более ограничен, чем у отрезков: целые числа, задающие наклон, не должны превосходить 4 по абсолютной величине (и по-прежнему не должны иметь общих делителей). Точной отсчета стрелки является ее начало.

3. Окружности, круги и овалы

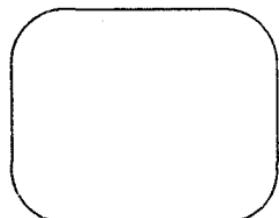
Окружность задается командой `\circle{}`, а круг (сплошной черный круг) — ее вариантом «со звездочкой» `\circle*`. У этих команд единственный аргумент — диаметр круга или окружности. Как обычно, он задается в единицах, равных значению параметра `\unitlength` (по умолчанию — в пунктах). Точной отсчета окружности или круга является центр. Вот пример картинки с окружностями и кругами:



```
\begin{picture}(100,80)
\put(30,30){\circle{30}}
\put(70,30){\circle{30}}
\put(30,50){\circle{30}}
\put(70,50){\circle{30}}
\put(50,40){\circle*{20}}
\end{picture}
```

Количество реально возможных диаметров кругов ограничено. Если окружности или круга с диаметром, указанным в качестве аргумента команды `\circle` или `\circle*`, в L^AT_EX'овских шрифтах нет, то будет напечатана окружность (круг), диаметр которой наиболее близок к указанному.

Наряду с окружностями и кругами, на псевдорисунок можно нанести также «oval» — прямоугольник с закругленными углами. Он задается командой `\oval`, аргументы которой — ширина и высота овала. Эти аргументы задаются в *круглых скобках* через запятую. Точка отсчета овала — его центр. Пример:



```
\begin{picture}(100,80)
\put(50,40){\oval(100,80)}
\end{picture}
```

Кроме того, возможны и «неполные» овалы, представляющие собой половины или четверти от полных. Чтобы задать такой неполный овал, надо задать команде `\oval` необязательный аргумент (в квадратных скобках, после обязательного). Для задания половины овала этот аргумент должен быть одной из следующих букв:

- t** Верхняя половина
- b** Нижняя половина
- r** Правая половина
- l** Левая половина

Для задания четверти овала необязательный аргумент команды `\oval` должен быть сочетанием двух из этих букв (например, `tr` для верхней правой четверти). Точка отсчета усеченного овала расположена там же, где точка отсчета соответствующего ему полного овала. Вот пример картинки с усеченными овалами:



```
\begin{picture}(100,80)
\put(50,40){\oval(80,60)[t]}
\put(50,40){\oval(80,60)[br]}
\end{picture}
```

4. Кривые (*L^AT_EX 2 _{ϵ}*)

При пользовании *L^AT_EX*'ом 2 _{ϵ} вы имеете возможность нанести на псевдорисунок кривую более или менее произвольной формы (эти кривые — так называемые квадратичные сплайны Безье). Это делается с помощью команды `\qbezier`. Вот пример ее работы:



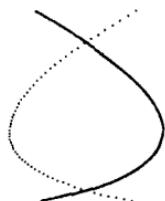
```
\begin{picture}(80,80)
\qbezier(22,2)(120,20)(20,77)
\put(22,2){\circle*{5}}
\put(120,20){\circle*{5}}
\put(20,77){\circle*{5}}
\end{picture}
```

После `\qbezier` надо указать (без пробелов, в круглых скобках, через запятую) координаты трех точек: начальной, «опорной» и конечной. Из начальной точки кривая выходит, устремляется к опорной, но, как правило, до нее не доходит, поскольку сворачивает к конечной точке, в которой и заканчивает свой путь. В нашем примере мы для ориентировки нанесли на псевдорисунок черные кружки с координатами в этих трех точках.

Никакой мистики в том, что *TeX* рисует кривые, нет: эти кривые просто составляются из сотен черных квадратиков. Соответственно, *L^AT_EX*'овское рисование кривых — процесс относительно медленный и занимающий весьма много машинной памяти. Если еще можно представить мучительно медленную работу с *L^AT_EX*'ом 2 _{ϵ} на слабом (скажем, AT286) компьютере, то рисование кривых Безье на такой технике — дело безнадежное.

Рисование кривых пойдет быстрее и отнимет меньше памяти, если попросить *L^AT_EX* не так густо ставить квадратики, из которых состо-

ит кривая. Для этих целей у команды `\qbezier` предусмотрен необязательный аргумент — количество этих квадратиков. Он ставится перед всеми обязательными в квадратных скобках:



```
\begin{picture}(80,80)
\qbezier(22,2)(120,20)(20,77)
\qbezier[60](58,2)(-40,20)(60,77)
\end{picture}
```

Кстати, обратите внимание, что опорная точка второй из наших кривых находится где-то за пределами текста. Это не страшно, поскольку ее координаты используются *Л^Т_EX*'ом только для расчетов.

Какой бы необязательный аргумент команды `\qbezier` мы ни задавали, количество квадратиков, из которых составляется кривая, не превысит числа 500. Если вы решили увеличить этот максимум, допустим, до тысячи, надо написать так:

```
\renewcommand{\qbeziermax}{1000}
```

Если так вы напишете в преамбуле, то предел 1000 будет относиться ко всем кривым в вашем тексте, а если внутри группы (например, внутри окружения), то изменение этого параметра забудется по выходе из группы. В гл. VII будет объяснено, что означает `\renewcommand` в общем случае.

5. Дополнительные возможности

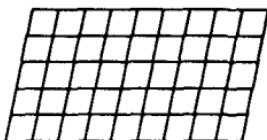
Иногда бывает нужно нанести на псевдорисунок несколько регулярно расположенных объектов. В этом случае, вместо того чтобы много раз писать `\put`, удобно воспользоваться командой `\multiput`. Она располагает на псевдорисунке несколько одинаковых объектов на равных расстояниях. Синтаксис этой команды таков:

```
\multiput(x,y)(Δx,Δy){n}{объект}
```

Здесь *x* и *y* — координаты первого из размещаемых объектов (как и в обычной команде `\put`), Δx и Δy — расстояния, на которые каждый следующий объект будет сдвинут относительно предыдущего по горизонтали и вертикали, *n* — количество объектов, которые надо разместить, и, наконец, *объект* — это, как и у команды `\put`, описание того, что мы размещаем на рисунке. Пример:

```
\begin{picture}(100,80)
\multiput(10,70)(8,-6){8}{%
{\circle*{3}}}
\end{picture}
```

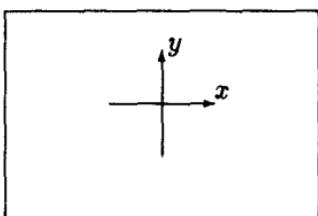
Обратите внимание на использование знака процента для удаления нежелательного пробела, создаваемого концом строки. Вот еще один пример; здесь с помощью команды `\multiput` рисуется решеточка:



```
\begin{picture}(100,50)
\multiput(0,0)(10,0){10}{%
{\line(1,5){10}}}
\multiput(0,0)(2,10){6}{%
{\line(1,0){90}}}
\end{picture}
```

С формальной точки зрения оба вышеприведенных примера совершенно правильны. Практически, однако, такое использование команды `\multiput` ведет к неоправданным затратам машинного времени. Например, каждый из наклонных отрезков во втором примере собирается из маленьких символов, причем TeX'у приходится повторять эту скучную операцию 10 раз. Разумнее было бы собрать этот отрезок лишь единожды, а дальше его просто копировать. TeX позволяет это сделать с помощью конструкции «блоковых переменных». Мы расскажем об этом в гл. VIII.

Иногда, когда псевдорисунок достаточно сложен, удобно применить следующий прием: задать в качестве аргумента одной из команд `\put` целое окружение `picture` (точкой отсчета будет служить левый нижний угол). При этом вы сможете отсчитывать координаты объектов на «подрисунке» относительно самого подрисунка, а не внешнего рисунка, что часто бывает проще; кроме того, если понадобится сдвинуть этот «подрисунок» как единое целое, то для этого будет достаточно изменить аргументы в одной-единственной команде `\put`. Вот пример рисунка с подрисунком (будем считать, что это классная доска, на которой нарисованы оси координат):



Этому рисунку соответствовал такой исходный текст:

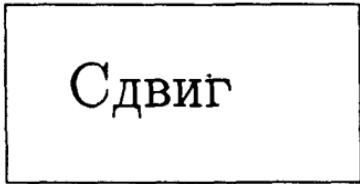
```
\begin{picture}(120,80)
% Края доски:
\put(0,0){\line(1,0){120}} \put(0,80){\line(1,0){120}}
\put(0,0){\line(0,1){80}} \put(120,0){\line(0,1){80}}
% Оси координат:
\put(40,25){\begin{picture}(40,40)%
    \put(20,0){\vector(0,1){40}}
    \put(0,20){\vector(1,0){40}}
    \put(40,22){$x$} \put(22,40){$y$}
\end{picture}}
\end{picture}
```

Кстати говоря, размеры внутренней картинки можно было бы задать совершенно произвольно, например, (200,200) или даже (0,0) — команда `\put` бездумно размещает объекты таким образом, чтоб их точки отсчета имели указанные координаты, и при этом не интересуется, сколько места они реально занимают и не наложатся ли на текст или другие объекты.

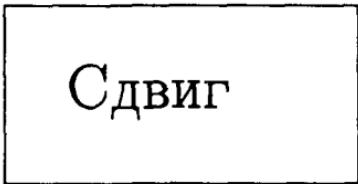
Нередко требуется сдвигать как единое целое не какую-то часть псевдорисунка, а весь псевдорисунок как целое (так приходится делать, когда вы ищете оптимальное расположение иллюстрации по отношению к тексту). Для этого удобно использовать еще одну возможность окружения `picture`: можно задать его таким образом, чтобы начало координат было не в левом нижнем углу, а в любой другой точке. Для этого после `\begin{picture}` надо задать не одну, а две пары чисел в круглых скобках. В этом случае первая пара чисел будет, как и прежде, обозначать ширину и высоту места, выделяемого \LaTeX'ом на псевдорисунок, а вторая пара будет указывать, каковы координаты левого нижнего угла этого псевдорисунка (по умолчанию, т. е. если второй пары чисел в круглых скобках нет, они были бы просто (0,0)). Главное только — не напутать со знаками: если вы сказали

```
\begin{picture}(a,b)(x,y)
```

то это значит, что левый нижний угол псевдорисунка имеет координаты (x, y) , стало быть, по сравнению со случаем, когда $x = y = 0$, весь псевдорисунок сдвинется на $-x$ по горизонтали и на $-y$ по вертикали! Если вы ничего не поняли, посмотрите на следующий пример, в котором второй псевдорисунок сдвигается на 40 единиц вправо и на 10 единиц вверх по отношению к первому:



Сдвиг



Сдвиг

```
\begin{picture}(150,80)
\put(0,0){\line(1,0){140}}
\put(0,70){\line(1,0){140}}
\put(0,0){\line(0,1){70}}
\put(140,0){\line(0,1){70}}
\put(25,30){\Huge Сдвиг}
\end{picture}\v[25pt]
\begin{picture}(150,80)(-40,-10)
\put(0,0){\line(1,0){140}}
\put(0,70){\line(1,0){140}}
\put(0,0){\line(0,1){70}}
\put(140,0){\line(0,1){70}}
\put(25,30){\Huge Сдвиг}
\end{picture}
```

6. Параметры оформления псевдорисунка

Про один из таких параметров мы уже говорили — это единица измерения длин на псевдорисунке, обозначаемая `\unitlength`.

В какой-то мере можно регулировать и толщину линий на наших псевдорисунках. Для этого предусмотрены команды `\thinlines` (тонкие линии) и `\thicklines` (толстые линии). По умолчанию стоит режим, в котором линии будут тонкими. Команды `\thicklines` и `\thinlines` можно давать не только в преамбуле, но и в самом тексте (в том числе и внутри окружения `picture`, так что можно регулировать, какие линии будут толстыми, а какие тонкими). Если одна из этих команд дана внутри группы, то по окончании группы ее действие прекращается (не забывайте, что любое окружение само по себе образует группу).

Кроме того, можно задать произвольным образом толщину вертикальных и горизонтальных (но не наклонных!) линий. Для этих целей служит команда `\linethickness`. У этой команды один обязательный аргумент — толщина линий, выраженная в $\text{\TeX}'$ овских единицах длины. Если мы скажем

```
\linethickness{2.5mm}
```

то все вертикальные и горизонтальные отрезки на псевдорисунке будут иметь толщину 2.5 мм, и такой же будет сторона квадратиков, из которых в $\text{\TeX}'$ е 2ε составляются кривые.

Глава VI

Верстка текста с выравниванием

При работе на пишущей машинке печать таблиц, состоящих из нескольких колонок, не вызывает особых проблем: все литеры имеют одинаковую ширину, поэтому, если нужно напечатать колонку слов, идущих одно под другим, достаточно каждый раз делать одинаковое количество пробелов. Однако полиграфические шрифты (в частности, используемые *TeX'ом*) являются, как правило, «пропорциональными» (каждая буква имеет свою ширину), а в этом случае добиться выравнивания в колонках сложнее.

В настоящей главе мы рассмотрим два основных способа, предstawляемых *TeX'ом* для верстки текста с выровненными колонками (например, таблиц). Начнем с менее мощного, но более простого — имитации табулятора.

1. Имитация табулятора

1.1. Элементарные средства

Табулятор имитируется в *TeX'е* с помощью окружения *tabbing*. При верстке таблиц с помощью этого окружения пользователь сам задает места, в которых должна начаться очередная колонка. Конкретно это выглядит так. При наборе первой строки этого окружения можно в любой момент поставить команду `\=` — она отмечает очередное место, с которого начинается новая колонка («позицию табулятора», как на пишущей машинке). *TeX* это место (расстояние от начала строки) запоминает. В дальнейшем можно с помощью команды `\>` «перескочить» к очередной позиции табулятора — текст, следующий после этой ко-

манды, будет набираться, начиная с позиции табуляции. Строки разделяются командой `\\"`. Рассмотрим это на примере:

начало	середина	конец
раз	два	три
раз	два	три
начинаем продолжаем заканчиваем		

```
\begin{tabbing}
начало\quad\=середина\\
\quad\=конец\\
раз\>два\>три\\
раз\> два\> три\\
начинаем\>
продолжаем\>
заканчиваем\\
\end{tabbing}
```

В первой строке мы задали две позиции табуляции двумя командами `\=` (на всякий случай мы разделили дополнительными пробелами слова в первой строке и, тем самым, наши позиции табуляции; отсюда команда `\quad`). Первая строка завершается командой `\\"`, а во второй строке мы уже начинаем пользоваться установленными позициями табуляции. Слово «раз» напечаталось с начала строки (каждая строка начинается с крайней левой позиции, если отсутствует команда вроде `\>`, задающая переход к новой позиции). Далее идет команда `\>` — «перейти на следующую позицию табуляции». И действительно, следующее после нее слово «два» начинается со второй позиции — как раз там же, где начиналось слово «середина». Перед словом «три» стоит еще одна команда `\>` — оно печатается с третьей позиции, как раз под словом «конец», с начала которого мы эту позицию и определили. Третья строка ничем не отличается от четвертой, хотя в исходном тексте между командами `\>` и словами стоят пробелы. Дело в том, что *пробелы после команд \> игнорируются*. Наконец, в четвертой строке слова при печати налезли друг на друга. Это и не удивительно: окружение `tabbing` исправно начинает очередную порцию текста с той позиции табуляции, которую мы ему укажем, но при этом отнюдь не проверяет, сколько места этот текст реально займет и не будут ли перекрываться колонки — за это целиком отвечает тот, кто текст готовит. Видимо, в данном случае следовало оставить побольше места при определении позиций табулятора (например, написать в первой строке `\qquad` вместо `\quad`).

Кроме установки дополнительных интервалов экспериментальным путем, есть и другой способ правильно приставить позиции табулятора. Именно, если закончить строку не командой `\\"`, а командой с суровым названием `\kill`, то эта строка не будет напечатана, но все позиции табулятора, установленные в ней, будут запомнены `LATEX`'ом, и их можно будет использовать в последующих строках. В приведенном выше примере можно было бы написать так:

начало	середина	конец
раз	два	три
начинаем продолжаем заканчиваем		

```
\begin{tabbing}
начинаем \=продолжаем \=
заканчиваем\kill
начало
\>середина
\>конец\\
\bf раз\>\it два\>три\\
начинаем\>
продолжаем\>
заканчиваем\\
\end{tabbing}
```

Для экономии места мы убрали из этой таблички лишнее «раз, два, три»; помимо этого, обратите внимание, что при установке позиций табулятора в первой (не печатающейся) строке мы сделали пробелы между концом слова и командой `\=` (иначе в последней строке слова бы опять слились: нам нужно, чтобы первая позиция табулятора не была впритык к концу слова «начинаем»). Заметьте также, что во второй строке мы убрали команды `\quad`; можно было бы их и оставить — на внешний вид таблицы это бы никак не повлияло, поскольку позиции табулятора уже установлены и лишние пробелы перед очередной командой `\>` никого не волнуют. По этой же причине мы не потрудились оставить пробелы между словами и `\>` в последней, предназначенной для печати строке «начинаем, продолжаем, заканчиваем». Наконец, обратите внимание и на то, как мы меняли шрифт в строке «раз, два, три»: слово «три» переключилось на обычный шрифт само собой. Это объясняется тем, что часть текста окружения `tabbing`, расположенная между двумя командами `\>` или `\=`, образует группу.

Внутри окружения `tabbing`, используется команда `\=`, которая, как мог заметить внимательный читатель, обычно имеет совсем другой смысл — постановка диакритического знака над буквой (см. таблицу на с. 92). Команды `\`` и `\'` также имеют внутри этого окружения особый смысл, о котором пойдет речь дальше. Поэтому, если внутри `tabbing` нам понадобился диакритический знак (скажем, над буквой е), то надо руководствоваться такой таблицей:

Внутри окружения `tabbing`

вместо	надо набирать
<code>\=e</code>	<code>\a=e</code>
<code>\`e</code>	<code>\a`e</code>
<code>\'e</code>	<code>\a'e</code>

1.2. Более сложные средства

Интервалы и разрывы между строками. Команда `\`` внутри окружения `tabbing` может иметь необязательный аргумент, действующий формально так же, как для этой команды, употребляемой внутри абзаца: если в квадратных скобках поставить длину (измеренную в воспринимаемых \TeX 'ом единицах, см. разд. I.2.10, или же какой-либо \LaTeX -овский параметр, значением которого является длина, например, `\medskipamount`), то после этой строки будет сделан дополнительный интервал, величина которого равна указанной длине. Имеет команда `\`*` и «вариант со звездочкой»: если написать `\`*` вместо `\``, то после строки, завершающей этой командой, начинать новую страницу будет запрещено. Команда `\`*` также может принимать необязательный аргумент. Он имеет тот же смысл, что и для соответствующей команды без звездочки.

Переустановка позиций табуляции. Команды `\=`, устанавливающие позиции табуляции, можно давать не только в первой строке. Сначала пример:

парочка позиций табуляции			
плюс	еще одна здесь:		
теперь	их	уже	три
Вторую	мы	сменим	и посмотрим:
где	эти	позиции	теперь

```
\begin{tabbing}
парочка \=позиций
\=табуляции\`\
\>плюс\>еще
одна здесь:\=\`\
теперь\>их\>
уже\>три\`\
Вторую \>мы\quad
\=сменим \>
и посмотрим:\`\
где\>эти\>
позиции\>теперь\`\
\end{tabbing}
```

Теперь опишем точно, как команда `\=` взаимодействует с `\>`. Внутри окружения `tabbing` в каждый момент \TeX 'у известно некоторое количество позиций табуляции, занумерованных подряд, от нуля до какого-то целого числа (не более двенадцати). При входе в окружение известна только позиция с номером нуль (это — всегда начало строки). Увеличиваться число известных позиций может за счет команды `\=`, используются позиции табуляции командой `\>`. Если команда `\=` встречается в строке *после* того, как использованы все известные позиции табуляции, то количество известных позиций табуляции увеличивается на 1, и

очередная позиция табуляции устанавливается в месте, куда попала `\=`. Если же `\=` встречается в строке *до* того, как все известные позиции табуляции израсходованы, то новых известных позиций не прибавляется, просто очередная по счету позиция табуляции заменяется на ту, которую задает команда `\=`.

Иногда бывает необходимо в пределах одной и той же таблицы временно перейти на новое расположение позиций табуляции, а затем вернуться к прежнему. Для этого используются команды `\pushtabs` и `\poptabs`. Первая из них запоминает расположение позиций табуляции; после этой команды можно позиции переустановить, пользоваться этими новыми переустановленными позициями ... — после команды `\poptabs` значения старых позиций табуляции будут восстановлены. Пример:

раз	два	три	четыре
гиппопотам	аллигатор		
раз	два		
три	четыре		
one	two	three	four
viens	divi	tr̄is	četři

```
\begin{tabbing}
раз\quad\=два\quad\=три\quad\=четыре\\
\pushtabs гиппопотам\quad\=аллигатор\\
раз\>два\>три\>четыре\\
\poptabs
one\>two\>three\>four\\
viens\>divi\>tr\@{i}\>v{c}etři\\
\end{tabbing}
```

Команды `\pushtabs` и `\poptabs` должны быть «сбалансированы»: каждой `\pushtabs`, запоминающей позиции табуляции, должна соответствовать вспоминающая их `\poptabs`. Если это условие не выполнено, вы получите сообщение об ошибке. Обратите также внимание, что знак долготы над буквой *i* в слове *tr̄is* мы поставили с помощью команды `\@{i}`.

Экзотика. Для полноты картины опишем некоторые изысканные возможности окружения `tabbing`.

Команда `\`` (внутри окружения `tabbing`) размещает текст таким образом, чтобы он не начался, а заканчивался у позиции табуляции. Сама эта команда позиций табуляции «не тратит»; просто весь текст, расположенный между `\>` или `\=` и `\``, размещается левее позиции табуляции, определяемой командами `\>` или `\=`. Таким способом можно верстать таблицы, в которых колонки выровнены по правому краю, а не по левому, как получается при обычном использовании `tabbing`. Вот пример:

слева	справа	\begin{tabbing}
à gauche	à droite	\hspace{3.5cm}\=\kill
links	rechts	слева\>справа\\
pa kreisi	pa labi	\a'a gauche\>\a'a droite\\

links\>rechts\\

pa kreisi\>pa labi\\

\end{tabbing}

Еще раз обратите внимание, что для постановки диакритического знака над буквой а нам пришлось писать \a' вместо \' (см. с. 184).

Команда \' внутри окружения tabbing прижимает весь текст строки, идущий после нее, к правому краю; между этой командой и командой, завершающей строку, не должно быть команд, использующих или устанавливающих позиции табуляции. Например, таблицу, у которой первая колонка выровнена по левому краю, а вторая — по правому (как в предыдущем примере), можно было бы задать так:

слева	справа	\begin{tabbing}
à gauche	à droite	слева\'справа\\
links	rechts	\a'a gauche\'\a'a droite\\
pa kreisi	pa labi	links\'rechts\\

pa kreisi\>pa labi\\

\end{tabbing}

Кстати, здесь нам вообще не понадобилось устанавливать позиции табуляции. Впрочем, смотрится эта таблица неважно.

Как мы уже отмечали, при начале новой строки текст начинается с нулевой позиции табуляции, т. е. с начала строки. Команда \+ позволяет изменить такое положение вещей: после этой команды при начале каждой новой строки текст будет начинаться не с нулевой, а с первой позиции табуляции (как если бы каждая последующая строка начиналась с команды \>). Если дать еще одну команду \+, то текст в последующих строках будет начинаться уже и не с первой, а со второй позиции табуляции, и т. д. Команда \- внутри окружения tabbing означает вовсе не место, где можно перенести слово (впрочем, команда с таким действием в этом окружении и не нужна): она действует противоположно команде \+. Наконец, команда \<, будучи употребленной в начале строки (в других местах ее употреблять нельзя), действует аналогично \-, но в пределах только этой строки (а не всех последующих, как \+ и \-). Следующий ниже пример иллюстрирует все эти изыски.

раз два три четыре
 два
 три
 четыре
 три
 четыре
 два
 раз два три четыре

```
\begin{tabbing}
раз \=два \=три \=\kill
раз\>два\>три\>четыре\+\\
два\+\\
три\+\\
четыре\\
\<три\\
четыре\-\-\\
два\-\-
раз\>два\>три\>четыре\\
\end{tabbing}
```

Описанные в этом разделе возможности окружения `tabbing` на практике используются редко, поскольку для верстки сложных таблиц в L^AT_EX'е есть более удобное средство — окружение `tabular`. Переидем к его описанию.

2. Верстка таблиц

При пользовании окружением `tabbing` вы должны самостоятельно следить, чтобы разные колонки не накладывались друг на друга. Можно, однако, передать эти заботы программе: Т_EX предоставляет возможности для верстки таблиц, в которых ширина колонок выбирается автоматически (по максимальной ширине их содержимого). В L^AT_EX'е для этих целей используются окружения `tabular` (для набора таблиц с текстом) и `array` (для набора таблиц из формул). Помимо автоматизированного определения ширины колонки, эти окружения дают возможность верстать разлинованные таблицы, таблицы, в которых некоторые записи охватывают несколько колонок, и т. д. В гл. II уже шла речь про окружение `array`; здесь мы подробно разберем, как работает `tabular`; все возможности этого окружения, о которых идет речь в этой главе, доступны и для `array`, и в последнем разделе главы мы дадим примеры их использования.

2.1. Простейшие случаи

Окружение `tabular` задает таблицу. Окружению необходимо задать обязательный аргумент — *пreamble таблицы*. Преамбула, помещаемая в фигурных скобках непосредственно после `\begin{tabular}`, представляет собой, в простейшем случае, последовательность букв, описывающих структуру колонок таблицы (по букве на колонку). Буквы эти могут быть такими:

- l означает колонку, выровненную по левому краю;
 r означает колонку, выровненную по правому краю;
 c означает колонку с центрированным текстом.

Между `\begin{tabular}` (с преамбулой) и закрывающей окружение командой `\end{tabular}` располагается собственно текст таблицы. В нем команда `\\"` разделяет строки таблицы, а знак `&`, называемый «амперсандом», разделяет колонки таблицы внутри одной строки (так что текст между двумя ближайшими амперсандами описывает «одну графу» таблицы). Пробелы в начале или конце «графы» таблицы игнорируются. Если вы прочли разд. II.3.2, то могли заметить буквальное совпадение с тем, что там написано про окружение `array`. Разница лишь в том, что содержимое граф таблицы обрабатывается в окружении `tabular` как текст, а в окружении `array` — как формулы. А теперь — первый пример:

Тип перечня	нумерация	<code>\begin{tabular}{lc}</code>
<code>itemize</code>	нет	Тип перечня & нумерация <code>\[5pt]</code>
<code>enumerate</code>	есть	<code>\tt itemize & нет\\</code>
<code>description</code>	нет	<code>\tt enumerate & есть\\</code> <code>\tt description & нет\\</code> <code>\end{tabular}</code>

Обратите внимание на две вещи. Во-первых, команда `\\"`, завершающая первую строку, дана с необязательным аргументом. Он задается так же и имеет тот же смысл, как если бы эта команда была внутри абзаца (с. 110) или окружения `tabbing` (с. 185): после строки вставляется дополнительный вертикальный промежуток (кстати, между строками таблицы, определенной с помощью окружения `tabular`, разрыва страницы *никогда* не происходит, так что в этом окружении у команды `\\"` варианта «со звездочкой» нет). Во-вторых, команда `\tt` всякий раз сменяла шрифт только в одной графе таблицы, не действуя на соседние. Это объясняется тем, что *графа таблицы образует группу*, так что любые изменения параметров (в том числе текущего шрифта), проведенные в одной графике, не влияют на остальные.

Прежде чем мы начнем говорить о более конкретных вещах, скажем о том, как окружение `tabular` взаимодействует с текстом вне его. Подобно окружению `picture`, оно не начинает печатать с новой строки и не завершает текущего абзаца. Вся таблица, порождаемая этим окружением, рассматривается `TeX`'ом как одна большая буква; если окружение `tabular` встретилось в середине абзаца, эта «буква» будет помещена

в строку (соседние строки раздвинутся, чтобы она поместились), и результат будет выглядеть некрасиво. Если такое размещение текста не входит в ваши планы, начинайте окружение `tabular` между абзацами (после пустой строки или команды `\par`). Удобно также бывает поместить окружение `tabular` внутрь окружения `center` или подобного ему: тогда `LATEX` сам позаботится о пробелах между таблицей и окружающим текстом.

Иногда бывает полезно знать, как расположена «большая буква», представляющая собой окружение `tabular`, по отношению к строке, в которой она оказалась. Ответ: ее середина идет вровень с низом строки (точнее, с «базисной линией» — см. гл. VIII); соответственно, на половинной высоте находится и точка отсчета этой «буквы». Пример:

слово А Б
 В Г слово

слово
`\begin{tabular}{rr}`
 А & Б \ В & Г
`\end{tabular}`
 слово

Можно также дать окружение `tabular` с необязательным аргументом `b`: тогда «буква», созданная окружением `tabular`, будет выровнена по нижней строке; необязательный аргумент `t` дает выравнивание по верхней строке:

А Б А Б
В Г А Б В Г
 В Г

слово
`\begin{tabular}{rr}`
 А & Б \ В & Г
`\end{tabular}` ·
`\begin{tabular}[t]{rr}`
 А & Б \ В & Г
`\end{tabular}`
`\begin{tabular}[b]{rr}`
 А & Б \ В & Г
`\end{tabular}`

Как мог заметить читатель, необязательный аргумент в данном случае ставится перед обязательным.

`LATEX` дает возможность сверстать и разлинованную таблицу. Для этого необходимо уметь задавать в таблице команды для создания горизонтальных и вертикальных отрезков («линеек» на полиграфическом жаргоне — см. разд. III.10). Горизонтальные отрезки задаются с помощью команды `\hline`. Эта команда может непосредственно следовать либо после `\backslash` (тогда отрезок печатается непосредственно после строки, завершенной этим `\backslash`), либо после `\begin{tabular}` и преамбулы (тогда отрезок печатается перед началом таблицы). Задаваемый командой `\hline` горизонтальный отрезок имеет ширину, равную общей ширине таблицы. Что касается вертикальных отрезков, то давайте для

начала также ограничимся случаем, когда эти отрезки, разделяющие колонки таблицы, простираются на всю ее высоту, сверху донизу. Такие отрезки проще всего предусмотреть в преамбуле таблицы. До сих пор мы говорили, что преамбула таблицы — это последовательность из букв `l`, `c` или `r`, характеризующих колонки. На самом деле в преамбуле может присутствовать и информация, описывающая то, что должно быть между колонками таблицы. В частности, символ `|`, помещенный в преамбулу таблицы между буквами, описывающими колонки, задает вертикальную линейку, разделяющую эти колонки. Можно поставить символ `|` перед первой из этих букв или после последней — тогда вертикальная линейка будет ограничивать таблицу слева или справа. Несколько таких символов могут стоять подряд — тогда колонки будут разделяться не одинарной, а двойной, тройной, и т. д. вертикальной линейкой. Вот пример разлинованной таблицы:

слон	<i>zilonis</i>
бегемот	<i>nilzirgs</i>
лев	<i>lauva</i>

```
\begin{tabular}{||l|l||}
\hline
слон & zilonis\\
бегемот & nilzirgs\\
лев & lauva\\
\hline
\end{tabular}
```

Две команды `\hline` могут следовать одна непосредственно за другой; в этом случае на печати получатся две горизонтальные линейки, одна под другой, разделенные по вертикали небольшим интервалом. Если слева и справа таблица ограничена вертикальными линейками, то на пересечении крайних вертикальных линеек с горизонтальными на печати получится разрыв:

Северо-Запад	Северо-Восток
Юго-Запад	Юго-Восток

```
\begin{tabular}{|c|c|} \hline
Северо-Запад & Северо-Восток\\ \hline
\hline
Юго-Запад & Юго-Восток\\ \hline
\end{tabular}
```

Позже мы расскажем, как можно от этого избавиться.

2.2. Более сложные случаи

Надписи, охватывающие несколько колонок. Чтобы создать такую надпись, нужно на месте соответствующей графы таблицы запи-

Таблица VI.1

Западные сладости		
Название	Количество	Цена
Сникерс	штука	330
	десяток	3000
Марс	штука	270
Баунти	штука	350
Твикс	В продаже	
Виспа	сегодня нет	

сать команду `\multicolumn`. У этой команды три обязательных аргумента:

- 1) Количество колонок, охватываемых нашей «нестандартной» графой.
- 2) «Преамбула» нашей графы. В качестве таковой может выступать буква l, r или c (если мы хотим, чтобы текст в графе был прижат влево, вправо или центрирован соответственно), возможно, с символами | слева или справа, если мы хотим, чтобы графа была ограничена вертикальными отрезками.
- 3) Текст, записываемый в графу.

Если таблица, в которой вы используете `\multicolumn`, является к тому же еще и линованной, то возможностей команды `\hline` для рисования горизонтальных отрезков может не хватить: иногда бывает нужен горизонтальный отрезок, простирающийся не на всю ширину таблицы, а охватывающий только часть ее колонок. Для рисования таких отрезков предусмотрена команда `\cline`. Как и `\hline`, ее нужно давать сразу после `\backslash`, но она имеет обязательный аргумент — номера первой и последней из колонок, охватываемых горизонтальной чертой, разделенные знаком «минус». Пример, в котором используются обе вышеописанные команды, приведен в таблице VI.1: Получается эта таблица таким образом:

```
\begin{tabular}{|l|l|r|}\hline&&\multicolumn{3}{|c|}{Западные сладости}\hline\hline Название & Количество & Цена\hline
```

Таблица VI.2

Я видел раков	
Вчера:	Сегодня:
Маленькие, но по три рубля, но очень маленькие, но по три, но очень маленькие.	Большие, но по пять рублей, но большие, но по пять ру- блей, но очень большие, но по пять.

```
\hline
Сникерс & штука & 330\\
\cline{2-3}
& десяток & 3000\\
\hline
Марс & штука & 270\\
\hline
Баунти & штука & 350\\
\hline
Твикс & В продаже & \\
\cline{1-1}
Виспа & сегодня нет & \\
\hline
\end{tabular}
```

Абзацы в графах таблицы. Иногда требуется, чтобы в графе таблицы стояла не строка, а абзац текста, переносы и разрывы строк в котором находятся автоматически. Чтобы этого добиться, надо в преамбуле вместо буквы l, с или r, описывающей структуру колонки, написать p{...}, где вместо многоточия должна быть указана ширина колонки (в ТЕХ'овских единицах — см. с. 25). Как можно представить в виде таблицы известную шутку М. М. Жванецкого, изображено на табл. VI.2, а вот так она задается в LATEX'e:

```
\begin{tabular}{|p{5cm}|p{5cm}|}
\hline
\multicolumn{2}{|c|}{\large\bf Я видел раков}\\
\hline
Вчера: & Сегодня: \\
Маленькие, но по три рубля, но очень  

маленькие, но по три, но очень маленькие.
\hline
\end{tabular}
```

&

Большие, но по пять рублей, но большие,
по пять рублей, но очень большие,
но по пять.\\"

```
\hline
\end{tabular}
```

Что такое колонка? При работе с линованными таблицами возникает следующий вопрос: как, собственно говоря, L^AT_EX понимает слова «одна колонка»? Пусть, например, преамбула таблицы имеет вид ||c|l||r|l|l, и мы в одной из строк написали, скажем,

Что-то&\multicolumn{1}{r}{Что-то еще}&&&И еще&Еще\\ .

Спрашивается, напечатается ли в этой графе вертикальный отрезок между первой и второй колонками? Другой пример: пусть в таблице с той же преамбулой какая-то из строк имеет вид

Слово & Еще слово & Еще одно\\

(стало быть, заканчивается эта строка преждевременно); спрашиваеться, сколько вертикальных отрезков будет напечатано в конце этой строки: два, один или ни одного? Ответ таков. L^AT_EX разделяет преамбулу на части, соответствующие колонкам. Если в преамбуле присутствуют только буквы l, c, r или p, то каждая такая часть — это просто соответствующая буква (p — вместе с выражением p{\dots}). Если же, кроме этого, в преамбуле присутствуют вертикальные черточки между буквами или так называемые at-выражения (о них речь пойдет ниже), разделение преамбулы на колонки происходит по таким правилам:

- В каждой из колонок присутствует одна и только одна из букв l, c, r или p (последняя — вместе со всем выражением p{\dots}).
- Каждая колонка, кроме первой, начинается с буквы.

В нашем примере, в частности, колонки устроены так:

Преамбула	c l r l l
Первая колонка	c
Вторая колонка	1
Третья колонка	1
Четвертая колонка	r
Пятая колонка	1
Шестая колонка	1

Таблица VI.3

	существительные формы	прилагательные формы
мой	{ le mien, la mienne les miens, les miennes	mon, ma, mes
твой	{ le tien, la tienne les tiens, les tiennes	ton, ta, tes
его, ее, свой	{ le sien, la sienne les siens, les siennes	son, sa, ses
наш	le notre, la notre, les notres	notre, nos
ваш	le votre, la votre, les vôtres	votre, vos
их, свой ¹	le leur, la leur, les leurs	leur, leurs

¹ Лишь в значении принадлежности 3-му лицу.

Поэтому в конце графы таблицы с такой преамбулой, оборванной после третьей колонки, будут напечатаны два вертикальных отрезка, поскольку они принадлежат третьей колонке. А если на месте второй графы такой таблицы написано `Что-то\multicolumn{1}{r}{Что-то еще}`, то вертикальный отрезок между первой и второй колонками также будет напечатана: этот отрезок является принадлежностью первой колонки, и команда `\multicolumn`, меняющая оформление второй колонки, отменить его не может.

3. Примеры

В этом разделе мы приведем различные примеры верстки сложных таблиц с помощью $\text{\LaTeX}'a$. По ходу дела будет рассказано и о некоторых изысканных возможностях окружений `tabular` и `array`, о которых до сих пор речи не было. Кое-где в этом разделе мы будем предполагать, что читатель знаком со средствами математического набора, описанными в гл. II.

Наш первый пример — таблица французских притяжательных местоимений, взятая из русско-французского словаря акад. Л. В. Щербы (табл. VI.3), которую мы задали в $\text{\LaTeX}'e$ так:

```
\small
\begin{tabular}{c|l}
\multicolumn{2}{c}{существительные формы} \\
& прилагательные формы\\[5pt]
мой & \$\left\{ \begin{array}{l} le mien, la mienne \\ les miens, les miennes \end{array} \right. \\
твой & \$\left\{ \begin{array}{l} le tien, la tienne \\ les tiens, les tiennes \end{array} \right. \\
его, ее, свой & \$\left\{ \begin{array}{l} le sien, la sienne \\ les siens, les siennes \end{array} \right. \\
наш & le notre, la notre, les notres \\
ваш & le votre, la votre, les vôtres \\
их, свой1 & le leur, la leur, les leurs
\end{tabular}
```

```

\begin{tabular}{l}
    le mien, la mienne\\
    les miens, les miennes\\
\end{tabular}
\right.

$\\
& mon, ma, mes\\[8pt]
твой & $\\left\\{
\begin{tabular}{l}
    le tien, la tienne\\
    les tiens, les tiennes\\
\end{tabular}
\right.

$\\
& ton, ta, tes\\[8pt]
его, ее, свой &
$\\left\\{
\begin{tabular}{l}
    le sien, la sienne\\
    les siens, les siennes\\
\end{tabular}
\right.

$\\
& son, sa, ses\\[8pt]
наш & le n\^otre, la n\^otre, les n\^otres
& notre, nos\\
ваш & le v\^otre, la v\^otre, les v\^otres
& votre, vos\\
их, свой$^1$\\
& le leur, la leur, les leurs
& leur, leurs\\
\cline{1-1}
\multicolumn{3}{l}{$^1$\rule{0pt}{11pt}\footnotesize
    Лишь в значении принадлежности 3-му лицу.}
\end{tabular}%
}

```

Разберем, как устроена эта таблица. Как явствует из ее преамбулы `c11`, она состоит из трех колонок, из которых левая центрирована, а две другие прижаты влево. Соответственно, три последние графы набраны совершенно бесхитростно. Заголовок таблицы сделан с помощью команды `\multicolumn`; от следующей колонки в этой строке она отделена

знаком &. Команда `\``, завершающая первую строку таблицы, имеет необязательный аргумент; это сделано, чтобы отодвинуть заголовок на 5 пунктов по вертикали от остальной части таблицы.

Рассмотрим теперь, как устроена вторая графа (начинающаяся с местоимения «мой»). Текст

$$\left\{ \begin{array}{l} \text{le mien, la mienne} \\ \text{les miens, les miennes} \end{array} \right.$$

образует в нашей таблице одну «запись» (часть таблицы, расположенную на пересечении графы и колонки). Для того чтобы получить фигурную скобку требуемого (и неизвестного нам заранее) размера, мы воспользовались командами `\left` и `\right`, применяемыми при наборе формул (см. разд. II.2.6). Так как эти команды вне формул использовать нельзя, нам пришлось оформить этот фрагмент текста как формулу. Между `\left\{` и `\right.` стоит, как водится, та формула, по размеру которой получается фигурная скобка, заданная командой `\left\{` — в нашем случае эта «формула» является фрагментом текста, задаваемым с помощью еще одного окружения `tabular` (с преамбулой 1). Команды `\``, завершающие первые три графы основной части таблицы, имеют необязательные аргументы, задающие дополнительные вертикальные пробелы после этих граф. Если не задавать этих необязательных аргументов, то фигурные скобки будут упираться друг в друга и портить вид таблицы. Конкретные величины дополнительных пробелов были подобраны опытным путем.

К местоимению «свой» в последней строке таблицы дана сноска. Знак сноски реализован нами опять же как математическая формула — верхний индекс 1 к «пустой формуле»; текст сноски реализован как графа таблицы, охватывающая все три колонки (с помощью команды `\multicolumn`). Команда `\footnotesize` задает размер шрифта, используемый в обычных сносках (см. разд. III.4). Линия, отделяющая сноsku от остальной части таблицы, реализована с помощью команды `\cline`. Наконец, посмотрим, как задана цифра 1 в самом тексте сноски. Вместо ожидаемого `1` написано вот что:

```
$^1$\rule{0pt}{11pt}
```

Как объясняется в разд. III.10, команда `\rule` задает в данном случае невидимый символ, занимающий по вертикали 11 пунктов и не занимающий места по горизонтали. Мы поставили этот невидимый символ в качестве подпорки: без нее горизонтальная черта соприкасалась бы с цифрой 1.

Таблица VI.4

Понедельник	8 ³⁰ –15	Обед	11–12
Вторник	12–19	Обед	15–16
Среда	10–17	Обед	12 ³⁰ –13 ¹⁵
Четверг	9–17	Обед	12–13
Пятница	11–16	Обед	—
Суббота	8–14	Обед	11–12

Остается заметить, что вся таблица в целом набрана мелким шрифтом, поскольку при шрифте нормального размера таблица не поместилась бы по ширине на страницу.

Следующий пример (табл. VI.4) — расписание работы некой химичистки. *Л**Т**Е**Х*'овский исходный текст для этого расписания выглядит так:

```
\begin{tabular}{l r@{-->}l@{\quad}l@{-->}l}
Понедельник & $8^{30}$ & 15 & 11 & 12 \\
Вторник & 12 & 19 & 15 & 16 \\
Среда & 10 & 17 & $12^{30}$ & $13^{15}$ \\
Четверг & 9 & 17 & 12 & 13 \\
Пятница & 11 & 16 & & \\
Суббота & 8 & 14 & 11 & 12
\end{tabular}
```

В преамбуле тут используется конструкция, с которой мы пока не встречались. Объясним, что она делает.

До сих пор мы говорили, что в преамбуле каждая колонка таблицы может обозначаться символом *l*, *c*, *r* или *p{...}*, а по краям или между колонками могут еще стоять вертикальные черточки *|*, обозначающие разделительные вертикальные линейки. Это, однако, — не вся правда. В качестве разделителя колонок (а также с краев) в преамбуле может быть использовано еще и так называемое «at-выражение»¹⁹: символ *@*, непосредственно после которого в фигурных скобках записан какой-то текст, возможно, с *Л**Т**Е**Х*'овскими командами. В таблице этот текст будет вставлен между соответствующими колонками во всех строках (если, разумеется, формат какой-то графы таблицы не был изменен командой *\multicolumn*). Мы использо-

¹⁹Мы выбрали для него такое название, поскольку официально символ *@* называется «коммерческое at»; неофициально его называют самыми разными именами, от «собаки» до «бламбы».

вали *at*-выражение трижды: два раза для вставки тире и один раз — для слова «Обед». Возникает вопрос, зачем нам понадобились команды `\quad` и `\quad` вокруг этого слова? Дело в том, что между колонками, разделенными *at*-выражением, не вставляется дополнительный интервал, которым *L^AT_EX* разделяет колонки в таблицах, созданных с помощью окружений `tabular` или `array`: именно поэтому тире между часом открытия химчистки и часом ее закрытия плотно прилегает к обоим числам. Слово «Обед», однако же, совсем не должно вплотную прилегать к началу обеденного перерыва, поэтому промежуток нужно создать самому, и проще это сделать один раз внутри все того же *at*-выражения, чем писать `\quad` шесть раз для каждого рабочего дня.

Иногда *at*-выражение имеет смысл применять даже в виде `\{ \}`: между колонками при этом ничего не вставится, но зато дополнительный интервал между колонками, разделенными этим выражением, будет подавлен. Если написать `\{ \}` в преамбуле перед символом, обозначающим первую колонку или после символа, обозначающего последнюю колонку, то будет подавлен дополнительный интервал, вставляемый перед первой или после последней колонки (иногда это помогает, если таблица немного не помещается на страницу по ширине).

Иногда интервал между колонками, автоматически устанавливаемый окружением `tabular` или `array`, является неудачным (ниже мы разберем соответствующий пример). В этом случае можно самостоятельно установить для него подходящее значение. Для этого надо присвоить новое значение параметру `\tabcolsep` для окружения `tabular` или `\arraycolsep` для окружения `array` (см. разд. I.2.6 по поводу параметров). По обе стороны от каждой колонки таблицы добавляется пробел размером `\tabcolsep` (соответственно, `\arraycolsep`). Стало быть, значение этих параметров — половина расстояния между соседними колонками.

Наряду с расстоянием между колонками можно менять толщину линеек в линованных таблицах (обозначается `\arrayrulewidth`; относится этот параметр как к `array`, так и к `tabular`), а также расстояние между соседними линейками — это расстояние обозначается `\doublerulesep`, и оно также относится в равной мере к `array` и к `tabular`.

Теперь разберем обещанный пример, в котором приходится менять заданное по умолчанию расстояние между колонками. Наш пример относится к делению многочленов в столбик. Посмотрите на такую фор-

муду:

$$\begin{array}{r} x^2 + 2x - 12 \\ x^2 + 5x \\ \hline - 3x - 12 \\ - 3x - 15 \\ \hline 3 \end{array}$$

Она была создана с помощью следующих L^AT_EX'овских команд:

```
$$
\arraycolsep=0.05em
\begin{array}{rrr@{\,\,}r|r}
x^2+2x&-12&&x+5\\
\cline{5-5}
x^2+5x&&&x-3\\
\cline{1-2}
&-3x&-12\\
&-3x&-15\\
\cline{2-3}
&&&3
\end{array}
$$
```

Сразу же скажем, зачем нам понадобилось менять `\arraycolsep`: без этого интервалы между слагаемыми в каждой строке выходили непомерно большими. А теперь разберем исходный текст подробнее. Начнем с преамбулы `rrr@{\,\,}r|r`. В ней первые три колонки отведены под слагаемые, наподобие x^2 , $+2x$ или -12 ; пятая колонка предназначена для делителя и частного ($x+5$ и $x-3$), а вертикальная черточка в преамбуле перед буквой `r`, задающей пятую колонку — для вертикального отрезка, входящего в состав «уголка». С другой стороны, в четвертой колонке нет вообще никакого текста: между третьим и четвертым знаками `&` ни в одной строке ничего не написано. Эту пустую колонку мы создали для того, чтобы вертикальный отрезок не пошел ниже, чем нужно: без нее, с преамбулой `rrr|r`, вертикальный отрезок относился бы к четвертой колонке (в соответствии с правилами на с. 194), и в результате третья строка закончилась бы вертикальным отрезком, что нам совсем ни к чему.

Осталось заметить, что пары долларов, ограничивающие выключную формулу, заодно ограничивают и группу, так что по окончании формулы закончится и группа, и старое значение `\arraycolsep` восстановится автоматически.

Наш последний пример использования окружения `tabular` связан с проблемой, с которой мы столкнулись на с. 191: как ликвидировать

разрыв в вертикальных линейках, получающийся, если в линованной таблице написать две команды `\hline` подряд? Первое, что приходит в голову, — создать еще одну графу в таблице, в которой поместить только невидимую линейку высотой, скажем, 2 пункта; казалось бы, тогда горизонтальные линейки будут на расстоянии 2 пункта друг от дружки, а вертикальные линейки не будут прерываться. Результат, однако, получается совершенно неудовлетворительный:

Северо-Запад	Северо-Восток
Юго-Запад	Юго-Восток

```
\begin{tabular}{|c|c|}
\hline
Северо-Запад & Северо-Восток\\
\hline
\rule{0pt}{2pt} & \\
\hline
Юго-Запад & Юго-Восток\\
\hline
\end{tabular}
```

Чтобы понять, в чем тут дело, нам придется обсудить, каким образом \LaTeX собирает таблицу из отдельных строк.

Таблицы, созданные с помощью окружения `tabular` или `array`, собираются из отдельных строк, которые вплотную приставляются друг к другу. При этом, чтобы расстояния между строками были одинаковыми, в каждую строку предварительно вставляется невидимая линейка (именно, линейка, создаваемая командой `\strut`). Из-за этой линейки расстояние между горизонтальными отрезками оказалось слишком большим, а наша линейка высотой в 2 пункта \LaTeX 'у не помогла: ведь `\strut` все равно выше! Чтобы обойти эту трудность, в \LaTeX 'е предусмотрен способ отменить автоматическую постановку `\strut`'ов во всех строках таблицы. Именно, для этого надо написать (*не* внутри окружения `tabular` или `array`) так:

```
\renewcommand{\arraystretch}{0}
```

Что такое `\renewcommand`, мы будем обсуждать в гл. VII, а пока давайте воспринимать этот рецепт догматически. Скажем только, что, во-первых, если эта команда была дана внутри группы, то по выходе из группы ее действие отменяется, и, во-вторых, в явном виде восстановление режима, когда в каждую строку таблицы вставляется `\strut`, достигается с помощью команды

```
\renewcommand{\arraystretch}{1} .
```

Теперь уже легко добиться желаемого эффекта: надо только не забыть поставить в нужные строки команду `\strut` в явном виде, коль скоро автоматически это теперь не делается. Итак, таблица

Северо-Запад	Северо-Восток
Юго-Запад	Юго-Восток

набирается следующим образом:

```
\renewcommand{\arraystretch}{0}%
\begin{tabular}{|c|c|}%
\hline
\strut Северо-Запад & Северо-Восток\\
\hline
\rule{0pt}{2pt}&\\
\hline
\strut Юго-Запад & Юго-Восток\\
\hline
\end{tabular}%
}
```

Знаки процента в конце некоторых строк мы поставили, чтобы концы этих строк не воспринимались как пробелы (на самом деле в данной ситуации вреда от пробелов не было бы). Закрывающая фигурная скобка в последней строке закрывает группу, из которой была дана команда `\renewcommand`.

Если граф в таблице много, то, возможно, вам не захочется много раз писать `\strut`. В этом случае можно включить эту команду в преамбулу с помощью `at-выражения`. Возможный вариант такой:

```
\renewcommand{\arraystretch}{0}%
\begin{tabular}{|c|c|}%
\hline
\strut\hspace{\tabcolsep} Северо-Запад & Северо-Восток\\
\hline
\multicolumn{1}{|c|}{\rule{0pt}{2pt}}&\\
\hline
\strut\hspace{\tabcolsep} Юго-Запад & Юго-Восток\\
\hline
\end{tabular}%
}
```

Если бы в аргументе `at-выражения` не был указан горизонтальный пробел размером `\tabcolsep`, то левая вертикальная линейка была бы напечатана вплотную к тексту (потому что `at-выражение` подавляет автоматически вставляемый горизонтальный пробел); заметим также, что теперь, когда `\strut` включен в `at-выражение`, нам пришлось воспользоваться командой `\multicolumn`, чтобы этот `\strut` не попал и в ту строку, где мы так старались от него избавиться.

Можно не только отменять автоматическое добавление $\backslash strut$ 'а в строки таблицы, но и изменять его высоту. Например, если мы хотим, чтобы размер этой линейки увеличился (во всех строках) в 3.7 раза, можно написать:

```
\renewcommand{\arraystretch}{3.7} .
```

Вместо десятичной точки можно поставить и десятичную запятую.

4. Дополнительные возможности $\text{\LaTeX}'a 2_{\varepsilon}$

Как водится, $\text{\LaTeX} 2_{\varepsilon}$ предоставляет дополнительные возможности по верстке таблиц. Возможности эти, однако, доступны не непосредственно, а только при подключении соответствующих стилевых пакетов. Ниже мы описываем два таких пакета, входящих в комплект поставки $\text{\LaTeX}'a 2_{\varepsilon}$.

4.1. Пересечения линеек

Возможностей окружения `array` вполне хватает для печати простейших линованных таблиц, но в более сложных случаях возникают проблемы (см. пример на с. 191). При пользовании $\text{\LaTeX}'om 2_{\varepsilon}$ можно подключить стилевой пакет `hhline`, облегчающий работу с линованными таблицами.

Итак, предположим, что этот пакет подключен (нелишне напомнить, что для этого достаточно после строки с командой `\documentclass` добавить команду `\usepackage{hhline}`). Тогда задания горизонтальных линеек становится доступной, наряду с уже известными `\hline` и `\cline`, новая команда `\hhline`, в аргументе которой описывается как сама линейка, так и ее пересечения с вертикальными линейками. Вот первый пример ее использования:

А	Б	В	Г
Д	Е	Ж	З

```
\begin{tabular}{|c|cc|c|}
\hline A & Б & В & Г\\
\hhline{|=|---|-|} 
Д & Е & Ж & З\\
\hline
\end{tabular}
```

Аргумент команды `\hhline` устроен следующим образом. Во-первых, в нем сказано, что на территории первой колонки линейка должна быть двойной (символ `=`), на территории второй колонки линейки не должно быть вовсе (символ `-` — «тильда»), а на территории третьей колонки линейка должна быть одинарной (символ `-`). Если в таблице n колонок, то в аргументе `\hhline` должны присутствовать n символов `-`, `=` или `-`, имеющих тот же смысл, что и выше.

Между этими символами, описывающими поведение линейки внутри колонок, расположены символы, описывающие пересечения горизонтальной линейки с вертикальными. В нашем примере это были вертикальные черточки |; кроме них, для задания информации о пересечениях линеек можно использовать символы :, #, а также буквы t и b. Какие именно пересечения линеек можно получить с их помощью, видно из следующей таблицы:

На печати		-	+, T, L		=	#, F, L
В аргументе \hhline	-	-	- -	=	=	= =
На печати	L	Г	Г, F, L	Л		
В аргументе \hhline	:=	=:	=::	-	-	- -
На печати	F	Г	ГГ	Г	#	#, F, L
В аргументе \hhline	:=	=:	=:::	#=	=#	=#=
На печати	F	L	ГГ	Г	F	L
В аргументе \hhline	t:=	b:=	=:t	=:b	=:t:=	=:b:=

Вот пример таблицы, в которой используются эти возможности команды \hhline:

1	2	3	4
5	6	7	8
A	B	V	G
D	E	Ж	З

```
\begin{tabular}{||cc||cc||}
\hhline{|t:::t:::t|} 
1 & 2 & 3 & 4 \\ 
\hhline{#:::::||} 
A & B & V & G \\ 
\hhline{||--||--} 
D & E & Ж & З \\ 
\hhline{|b:::b:::b|} 
\end{tabular}
```

Еще раз подчеркнем, что команда \hhline обрабатывает пересечения линеек независимо от того, какие вертикальные линейки заданы в преамбуле. Забота о том, чтобы аргумент \hhline был согласован с преамбулой, лежит на вас.

4.2. Расширение возможностей array и tabular

В этом разделе мы рассказываем о различных мелких (но нередко полезных) дополнительных возможностях, открывающихся при подключении стилевого пакета array.

Итак, предположим, что этот пакет подключен. Что нового вы сможете сделать?

При пользовании командой `\hline` горизонтальные линейки иногда слишком плотно примыкают к тексту (особенно если текст содержит прописные буквы). Если пользоваться средствами *ЛАTeX'a* 2.09, то для борьбы с этим надо либо писать

```
\renewcommand{\arraystretch}{...}
```

либо вставлять в каждую строку по дополнительной распорке. При подключении пакета `array` появляется и более простой способ: надо присвоить ненулевое значение параметру `\extrarowheight`. Это — величина, которая добавляется к высоте каждой строки таблицы. Этому параметру можно присваивать значения так же, как и любому другому параметру со значением длины (см. с. 25); по умолчанию его величина равна нулю, для лучшего отделения линеек от текста хорошо присвоить ему значение 2–3 пункта.

Если вы пользовались окружением `tabular` с необязательным аргументом `t`, задающим выравнивание таблицы как «буквы» по верхней строке, то могли обратить внимание, что это выравнивание нарушается, если таблица начинается с горизонтальной линейки:

а	б
в	г

а б
в г

```
\begin{tabular}[t]{rr}
а & б \\ в & г
\end{tabular}
\begin{tabular}[t]{|rr|} \hline
а & б \\ в & г \\ \hline
\end{tabular}
```

Чтобы выравнивание происходило не по линейке, а по первой строке текста, надо задать верхнюю линейку командой `\firsthline`, а не `\hline` — тогда получится то, что изображено на рис. VI.1. Аналогично, чтобы при пользовании `tabular` с необязательным аргументом `b` выравнивание таблицы как целого шло по нижней строке текста, а не по нижней линейке, надо нижнюю горизонтальную линейку задать командой `\lasthline`, а не `\hline`.

Остальные дополнительные возможности пакета `array`, о которых пойдет речь, относятся не к семантике, а к синтаксису. Иными словами, речь пойдет не о новых, недоступных ранее полиграфических возможностях, но о том, как более коротким путем добиться эффектов, доступных и в *ЛАTeX'e* 2.09.

Как мы знаем, в *ЛАTeX'e* 2.09 в преамбуле окружения `tabular` (а также `array`) могли стоять буквы `l`, `r`, `s` или выражение `p{...}`, обс-

а	б	в	г
д	е	ж	з

а	б	в	г
д	е	ж	з

а	б	в	г
д	е	ж	з

а	б	в	г
д	е	ж	з

```
\begin{tabular}[t]{rrrr}
а & б & в & г \\
д & е & ж & з
\end{tabular}

\begin{tabular}[t]{|rrrr|}\hline
а & б & в & г \\
д & е & ж & з\\ \hline
\end{tabular}\|[1in]

\begin{tabular}[t]{rrrr}
а & б & в & г \\
д & е & ж & з
\end{tabular}

\begin{tabular}[t]{|rrrr|}\firsthline
а & б & в & г \\
д & е & ж & з\\ \hline
\end{tabular}
```

Рис. VI.1.

значающие тип колонки, а между ними — вертикальные черточки или `at`-выражения. Пакет `array` добавляет кое-что к этому списку.

Во-первых, при подключении этого пакета в преамбуле, наряду с выражением `r{...}`, можно пользоваться выражениями `m{...}` и `b{...}`. Как и `r{...}`, они указывают, что в колонке стоит абзац текста ширины, заданной в фигурных скобках. Однако в графах абзац, заданный с помощью `b{...}`, выравнивается по своей нижней строке, абзац, заданный с помощью `m{...}` — по середине своей высоты, а абзац, заданный с помощью `r{...}`, всегда, в том числе и в `ЛATEX`’е 2.09, выравнивался по своей верхней строке (даже если вы об этом и не подозревали). Например, следующая «таблица»

Настроение

бодрое, идем
ко дну.

Настроение
бодрое, идем
ко дну.

Настроение
бодрое, идем
ко дну.

соответствует такому исходному тексту:

```
\begin{tabular}{b{.9in}m{.9in}p{.9in}}
Настроение бодрое,  
идем ко дну. &  
Настроение бодрое,

```

```
идем ко дну. &
Настроение бодрое,
идем ко дну.
\end{tabular}
```

Наряду с *at*-выражениями, пакет *array* позволяет использовать в преамбуле еще и *!-выражения*. Именно, между буквами, обозначающими колонки, можно, наряду с вертикальными черточками и *at*-выражениями, написать *!{...}*, где на месте точек стоят какие-то *TeX*'овские команды и/или текст. Эта конструкция оказывает то же действие, что и *at*-выражение, но при этом, в отличие от *at*-выражения, не подавляет интервал между колонками. Поэтому *!-выражение* удобно использовать для увеличения интервала между колонками: в таблице с преамбулой

```
{rc!{\vspace{2pt}}c1}
```

интервал между двумя центрированными колонками будет увеличен на два пункта.

Другое возможное применение *!-выражений* — печать линованных таблиц, в которых вертикальные линейки, разделяющие колонки, имеют разную ширину. Если, например, мы хотим, чтобы какая-то из вертикальных линеек имела ширину *1pt*, а не *\arrayrulewidth*, надо в преамбуле вместо вертикальной черточки *|*, обозначающей эту линейку, написать *!{\vrule width 1pt\relax}* (см. с. 139 по поводу команды *\vrule*).

Наконец, еще одна интересная возможность, предоставляемая пакетом *array*, — это автоматическая вставка *TeX*'овских команд в начале и/или конце колонки. Именно, если в преамбуле непосредственно перед любой из букв *l*, *c*, *r*, *t*, *m* или *b*, обозначающих тип колонки, вставить выражение

```
>{команды}
```

то *команды* будут автоматически добавляться в начало соответствующей колонки. Это может пригодиться, если вам нужно сменить шрифт в одной из колонок:

Людовик XIII	<i>1610–1643</i>
Людовик XIV	<i>1643–1715</i>
Людовик XV	<i>1715–1774</i>

```
\begin{tabular}{|l|>{\s1}l|}
\hline
Людовик XIII & 1610--1643\\
Людовик XIV & 1643--1715\\
Людовик XV & 1715--1774\\
\hline
\end{tabular}
```

Можно также после буквы, обозначающей тип колонки, или после конца `p-`, `m-`, или `b-` выражения, написать

`<{команды}`

чтобы `команды` были добавлены в конец колонки. Эта конструкция полезна, если какие-то из колонок в таблице, оформленной как `tabular`, должны набираться в математическом режиме — достаточно поставить в начале и конце этой колонки по знаку доллара. Пример:

квадрат суммы	$(x + y)^2$
квадрат разности	$(x - y)^2$
сумма кубов	$x^3 + y^3$

```
\begin{tabular}{l>{$}l<{$}}
квадрат суммы & (x+y)^2\\
квадрат разности & (x-y)^2\\
сумма кубов & x^3+y^3
\end{tabular}
```

Глава VII

Создание новых команд

Средства $\text{\LaTeX}'a$, описываемые в этой главе, позволяют значительно сократить число нажатий на клавиши при наборе сложных текстов. Именно, мы расскажем, как создавать новые команды (или, если угодно, сокращенные обозначения), заменяющие собой длинные фрагменты из текста и $\text{\TeX}'овских$ команд. Официально такие новые команды называются макроопределениями, а в разговорной речи — макросами или макро.

1. Макроопределения

1.1. Команды без аргументов

Начнем с примера. Пусть вы пишете текст, в котором регулярно встречается математический значок $\stackrel{\text{def}}{=}$ (он означает «равно по определению»). Пользуясь тем, что вы узнали из гл. II, нетрудно понять, что генерируется этот значок внутри математической формулы такой последовательностью команд:

```
\stackrel{\rm def}{=}
```

Часто писать такой длинный набор команд утомительно. Вот бы в $\text{\LaTeX}'e$ была предусмотрена команда, скажем, \eqdef , генерирующая символ бинарного отношения $\stackrel{\text{def}}{=}$! Правда, такой команды нет, но мы ее можем создать. Для этого следует написать так:

```
\newcommand{\eqdef}{\stackrel{\rm def}{=}}
```

После того как \TeX прочтет эту строку, он всюду, встречая команду \eqdef , будет реагировать точно так же, как если бы он видел текст $\stackrel{\rm def}{=}$. Например, формула $x^2 \stackrel{\text{def}}{=} x \cdot x$ теперь получается так:

```
x^2\eqdef x\cdot x
```

Новая команда $\text{\TeX}'$ а, которую мы определили, называется макросом (еще говорят: макроопределение, макрокоманда, макро). Рассмотрим точные правила для создания макросов средствами $\text{\LaTeX}'$ а.

Для создания макросов используется команда `\newcommand`. Эта команда имеет два обязательных аргумента. Первый из них — имя, которое вы придумали для вашего макроса. Имена макросов должны подчиняться тем же правилам, что имена $\text{\TeX}'$ овских команд (см. разд. I.2.3): либо `backslash` и после него одна не-буква, либо `backslash` и после него — последовательность букв. Второй обязательный аргумент команды `\newcommand`, называемый «замещающим текстом», сообщает $\text{\TeX}'$ у смысл макроса: на этот текст ваш макрос будет замещаться в процессе трансляции (как говорят, макрос будет «разворачиваться»).

При пользовании командой `\newcommand` нельзя в качестве имени макроса выбирать имя уже существующей команды или окружения. Впрочем, если вы и попробуете так сделать, \LaTeX вам этого не позволит, выдав сообщение об ошибке. Если ваша русификация $\text{\TeX}'$ а позволяет использовать в именах макросов русские буквы, имеет смысл этим воспользоваться: при этом резко снизится вероятность нарваться на запрещенное имя команды, не говоря уж о том, что русские имена более мнемоничны. Мы в дальнейших примерах также будем использовать русские буквы в именах макросов.

Еще одно ограничение: имя новой команды не должно начинаться на `end`.

Если команда `\newcommand` дана внутри группы, то смысл определяемой ею новой команды будет забыт $\text{\TeX}'$ ом по выходе из группы. Если новая команда определяется в преамбуле, то, естественно, она будет понята $\text{\TeX}'$ у на протяжении всего документа.

В «замещающем тексте» макроопределения не следует пользоваться командой `\newcommand` (или `\renewcommand`, о которой пойдет речь ниже). И еще одно ограничение, которое надо иметь в виду, создавая новые команды (макросы): во втором параметре команды `\newcommand` (иными словами, в «замещающем тексте») вместе с каждой открывающей фигурной скобкой должна присутствовать соответствующая ей закрывающая²⁰: «несбалансированных» фигурных скобок в замещающем тексте быть не должно, так что макроопределения наподобие

```
\newcommand{\начатькурсив}{\it}
\newcommand{\кончитькурсив}{\rm}
```

²⁰Фигурные скобки, входящие в состав команд `\{` и `\}`, в счет при этом не идут.

являются незаконными и приведут лишь к сообщению об ошибке. Если вам кажется, что такие ограничения стеснительны, можете изучить по книге [3], как их обходить; учите, однако, что тогда вам придется стать \TeX ником. Для большинства практических целей возможности по созданию макроопределений, предоставляемые $\text{\LaTeX}'ом$, вполне достаточны.

Еще одно ограничение: в замещающем тексте макроопределения нельзя пользоваться командой `\verb` или окружением `verbatim`.

Давайте теперь разберем несколько примеров, обращая внимание на типичные ошибки. Макросы хороши как средство скорописи, если вам приходится сталкиваться со сложными агрегатами $\text{\TeX}'овских$ команд в математических формулах. Их можно использовать и для совсем бесхитростных целей. Например, если в вашем тексте часто встречается знак Δ , то вам может надоест все время писать длинную команду `\bigtriangleup`. Коли так, придумайте сокращенное обозначение (скажем, `\btu`), напишите в преамбуле

```
\newcommand{\btu}{\bigtriangleup}
```

и вы сможете писать формулы наподобие

$$(A \Delta B) \cap C = (A \cap C) \Delta (B \cap C) \quad \$ (A \backslash \btu B) \cap C = \\ (A \cap C) \backslash \btu (B \cap C) \$$$

На с. 47 было рассказано, что делать, чтобы создать согласующееся с нашими традициями обозначение для функции тангенс. Теперь мы понимаем, что это был пример макроопределения (если вам интересно знать, что значит команда `\mathop` в замещающем тексте, загляните в разд. II.4.4).

В последних разделах гл. II вы найдете массу других примеров громоздких конструкций, для которых имеет смысл создать макросы. Практически польза от макроса начинает ощущаться, если конструкция, которую он заменяет, встречается в тексте не меньше четырех-пяти раз.

А вот пример типичной ошибки. Пусть в тексте, который вы набираете, регулярно встречаются фразы наподобие

Подмногообразия проективного пространства P^n — основной объект изучения алгебраической геометрии.

и пусть для сокращения письма вы написали в преамбуле

```
\newcommand{\Pn}{\$ {\bf P}^n \$}
```

Теперь можно писать, например, так:

... пространства $\mathbf{\mathcal{P}_n}$ --- основной объект...

Однако для набора формулы $x \in \mathbf{P}^n$ написать $\$x\backslash in\mathbf{P}n\$$ не удастся: появится сообщение о том, что символ \backslash преступно употреблен вне математической формулы. Удивительного в этом нет: TeX исправно подставляет вместо $\mathbf{P}n$ тот «замещающий текст», который вы ему сообщили во втором аргументе команды `\newcommand`. В результате этого при развертывании макроса $\mathbf{P}n$ текст $\$x\backslash in\mathbf{P}n$$ превращается в незаконный текст $\$x\backslash in \$\{\mathbf{bf}\ P\}^n$$, в котором математическая формула заканчивается со вторым из знаков доллара, а символ \backslash оказывается посреди обычного текста. Чтобы можно было напечатать \mathbf{P}^n не только изолированно, не надо включать знаки доллара в определение:

`\newcommand{\mathbf{P}_n}{\{\mathbf{bf}\ P\}^n}`

При этом придется, конечно, ставить знаки доллара вокруг $\mathbf{P}n$ в тех случаях, когда в тексте встречается просто \mathbf{P}^n , но зато наш макрос можно будет использовать и как составную часть более сложных формул.

- ε ИTeX 2 ε предоставляет и другой способ борьбы с этой проблемой: определите $\mathbf{P}n$ как

`\newcommand{\mathbf{P}_n}{\ensuremath{\{\mathbf{bf}\ P\}^n}}`

(без всяких знаков доллара) — и вы сможете спокойно пользоваться своей новой командой $\mathbf{P}n$ как в тексте, так и в формулах:

Пусть \mathbf{P}_n --- n -мерное проективное пространство, и пусть $X \subset \mathbf{P}n$ --- неприводимое многообразие...

- (знаки \backslash мы поставили, чтобы строчка не смогла начаться с типа — с. 89). Команда `\ensuremath`, определенная в ИTeX'e 2 ε , всегда обрабатывает свой аргумент как математическую формулу, независимо от того, в тексте или в формуле вы ее используете. Впрочем, пока ИTeX 2 ε окончательно не завоевал мир, этой командой лучше не увлекаться: если в файле для ИTeX'a 2 ε используются макросы, определенные с участием `\ensuremath`, то приводить его ε к виду, понятному для ИTeX'a 2.09, довольно утомительно.

Создавать макросы полезно не только для сокращения числа нажатий на клавиши при наборе формул. Вот пример, когда макросы помогают и при наборе обычного текста. Предположим, в нашем тексте много задач, причем условие каждой из задач начинает новый абзац (как

обычно и бывает). Предположим также, что эти задачи никак не нумеруются (это предположение менее реалистично, но оно сделано только для простоты и временно; в дальнейшем мы узнаем, как сделать так, чтобы \LaTeX сам нумеровал задачи для нас). Слово «Задача», с которого начинается условие, хочется как-то выделить в тексте; предположим, мы решили выделять его жирным шрифтом. Давайте создадим макрос, который будет делать все это за нас, чтобы можно было не печатать каждый раз $\{\text{\bf}\dots\}$, а просто написать \z . Первым обычно приходит в голову что-нибудь такое:

```
\newcommand{\z}{\bf Задача}
```

Посмотрите, что из этого выйдет:

Задача. Пять парней за пять дней съели пять окуней. За сколько дней пятнадцать парней съедят пятнадцать окуней?

Почему же жирным шрифтом напечаталось не только слово «Задача», но и весь дальнейший текст? Ответ: \TeX опять пунктуально заменил \z на «замещающий текст», в результате чего получилось вот что:

\bf Задача. Пять парней ...

Команда \bf оказалась *не* внутри группы, и весь текст пошел жирным шрифтом. Чтобы не попадаться в такую ловушку, надо помнить, что при развертывании макроса фигурные скобки, ограничивающие замещающий текст в команде \newcommand , отбрасываются. Правильнее было бы дать такое определение:

```
\newcommand{\z}{\{\bf Задача\}}
```

На сей раз \z будет заменяться на $\{\text{\bf Задача}\}$, чего мы и хотели (при развертывании макроса отбрасывается внешняя пара фигурных скобок, ограничивающая второй аргумент команды \newcommand , и только она!).

ε При пользовании $\text{\LaTeX}'ом$ 2ε лучше написать (см. с. 99):

```
\newcommand{\z}{\{\normalfont\bfseries Задача\}}
```

Впрочем, наш макрос \z еще не идеален. Во-первых, после слова, напечатанного жирным шрифтом, следует поставить жирную точку, поэтому точку тоже разумно включить в макроопределение. Далее, согласно общему правилу игнорирования пробелов после имени команды, состоящего из букв, в тексте при этом нельзя будет написать

$\text{\z}.$ Пять парней за пять дней съели пять окуней. За сколько дней пятнадцать парней съедят пятнадцать окуней?

`\z Пять парней...`

так как при этом пропадет пробел между словом «Задача» и следующим словом. Можно этот пробел, как водится, всякий раз специально организовать, написав

`\z{} Пять парней...`

но лучше вставить и пробел прямо в определение:

`\newcommand{\z}{\bf Задача. }{}`

Теперь запись `\z Пять парней...` даст то, что нужно. Впрочем, если угодно, вот еще один штрих. Если вы в какой-нибудь момент забудете оставить пустую строку перед очередной задачей, то слово «Задача» при этом будет не начинать абзац, а продолжать предшествующий текст. Чтобы этого раз и навсегда избежать, можно вставить команду «закончить абзац» в наше макроопределение. Как вы помните, эта команда называется `\par` (с. 117):

`\newcommand{\z}{\par\bf Задача. }{}`

Если перед нашим макросом `\z` пустая строка все-таки будет, то лишняя команда `\par` ни на что не повлияет (см. с. 117 и ниже), а если ее не будет, то `\par` сыграет роль недостающей пустой строки.

После того как какой-то элемент текста оформлен с помощью макроса, становится удобно и менять это оформление. Например, если вдруг понадобилось оформлять все задачи так, чтобы соответствующие абзацы начинались без отступа, достаточно внести небольшое изменение в наше определение команды `\z`:

`\newcommand{\z}{\par\noindent\bf Задача. }{}`

Если бы каждую задачу мы оформляли вручную, нам пришлось бы перед каждым словом «Задача» вписывать команду `\noindent`. В дальнейшем, познакомившись с L^AT_EX'овскими «счетчиками», мы усовершенствуем наш макрос `\z` таким образом, чтобы он еще и автоматически нумеровал задачи.

Мы уже говорили, что переопределить значение уже существующей команды с помощью `\newcommand` невозможно. Иногда, однако, это бывает необходимо. Пример тому приведен в гл. IV: если мы хотим, чтобы в заголовках глав печаталось именно слово «Глава», а не «Chapter», то необходимо определить по-новому команду `\chaptername`. Для такого рода целей используется команда `\gnewcommand`. Она устроена точно так же, как `\newcommand`, с тем отличием, что в качестве ее первого аргумента надо указывать имя уже существующей команды; в

в этом случае по выполнении команды `\newcommand` значение этой команды изменится: она превратится в сокращенное обозначение для текста, указанного в качестве ее второго аргумента. Например, если написать

```
\renewcommand{\alpha}{Ку-ку}
```

то команда `\alpha` будет генерировать не то, что обычно (букву α в математической формуле и сообщение об ошибке, если эта команда употреблена вне математической формулы), а текст «Ку-ку».

Если команда `\renewcommand` дана внутри группы, то созданное ею переопределение значения команды забудется по выходе из этой группы. Если в качестве первого аргумента команды `\renewcommand` указать имя несуществующей команды, то вы получите сообщение об ошибке.

Команда `\renewcommand` при неаккуратном обращении может привести к неприятностям. Дело в том, что нередко команды $\text{\LaTeX}'$ определяются в терминах других команд $\text{\LaTeX}'$, причем знать эти сложные связи рядовой пользователь вовсе не обязан. На практике такое незнание может привести к тому, что безобидное на первый взгляд использование в своих целях имени какой-нибудь команды, которая в вашем тексте ни разу не встречается, вызовет необъяснимо неправильную работу других команд. Поэтому используйте `\renewcommand` только тогда, когда вы полностью отдаете себе отчет в своих действиях.

Выше мы часто называли макросы просто командами, и это — не просто небрежность речи: на самом деле принципиальной разницы между макросами и $\text{\TeX}'$ овскими командами почти нет. В частности, почти все команды $\text{\TeX}'$, о которых говорилось и еще будет говориться в нашей книге, являются именно макросами; в отличие от макросов, создаваемых нами с помощью `\newcommand`, их смысл уже известен $\text{\TeX}'$ у к моменту запуска программы, так что определять их каждый раз не нужно. Макросы, в свою очередь, могут ссылаться на другие макросы (кстати, команда `\stackrel` на которую ссылалась наша команда `\eqdef`, также является макросом) ... — и так далее, пока не дойдет до так называемых «примитивных команд» $\text{\TeX}'$ а. Из команд, о которых мы вам рассказывали, примитивными являются считанные единицы: `\left`, `\right`, `\noindent` и некоторые другие.

1.2. Команды с аргументами

В предыдущем разделе мы научились создавать новые команды, не требующие аргументов. Но мы хорошо знаем, что многие $\text{\TeX}'$ овские команды принимают аргументы, и часто возникает потребность создать новую команду с такими возможностями. Пусть, например, в вашем тексте часто встречается «символ Лежандра», выглядящий так: $\left(\frac{a}{p}\right)$. Для получения этого символа в исходном тексте надо написать (внутри формулы, естественно) так:

```
\left(\frac{a}{p}\right)
```

Хорошо бы создать команду `\символ` с двумя аргументами, чтобы можно было написать в формуле `\символ{a}{p}` и получить на печати $\left(\frac{a}{p}\right)$. Что ж, L^AT_EX предоставляет нам возможность сделать и это. Для создания команды с аргументами используется все та же команда `\newcommand`, но с необязательным аргументом. Посмотрите, как можно определить команду `\символ`:

```
\newcommand{\символ}[2]{\left(\frac{#1}{#2}\right)}
```

Разберем, что означает эта запись. В квадратных скобках стоит количество аргументов в нашем макросе (в нашем случае 2). Далее, в самом «замещающем тексте» появились значки `#1` и `#2`. При развертывании макроса на их место будут подставляться соответственно первый и второй аргументы нашей новой команды `\символ`. Например, если в формуле написать

```
\символ{a+b}{c}
```

то будет напечатано $\left(\frac{a+b}{c}\right)$.

Теперь рассмотрим точные правила. Необязательный аргумент команды `\newcommand`, который должен быть расположен между двумя обязательными, указывает, сколько аргументов будет требовать создаваемая вами команда (макрос). Это количество аргументов в макросе с аргументами не может быть более 9. В «замещающем тексте» места, на которые при развертывании макроса будут подставляться аргументы, обозначаются символами `#1` для первого аргумента, `#2` для второго аргумента и т. д. Эти символы могут идти в любом порядке и присутствовать любое количество раз (в том числе и ни разу). Когда мы будем использовать нашу новую команду в тексте, после ее имени в фигурных скобках должны будут следовать аргументы, ровно в том количестве, которое мы указывали в необязательном аргументе команды `\newcommand`, и каждый — в своей паре фигурных скобок (как обязательные аргументы любой другой L^AT_EX'овской команды). При развертывании макроса на место его и его аргументов будет подставлен «замещающий текст», в котором вместо `#1` всюду стоит первый аргумент, вместо `#2` — второй аргумент, и т. д.

Проиллюстрируем все сказанное примером. Пусть мы определили команду `\путаница` следующим образом:

```
\newcommand{\путаница}[4]{K#4k#2#1l}
```

Тогда будет получаться, например, такое:

Крокодил Гена и его друзья. \путаница{ди}{о}{го}{ро}
Гена и его друзья.

В самом деле, первым аргументом команды \путаница будет ди, вторым о, третьим го, четвертым ро; при развертывании появится сначала буква К, затем четвертый аргумент, затем буква к, затем второй, первый и наконец буква л — как если бы Крокодил присутствовал в исходном тексте собственной персоной.

Ни аргументы L^AT_EX'овских команд (в том числе и определенных вами), ни «замещающий текст» в \newcommand не должны содержать «несбалансированных» (не имеющих пары) фигурных скобок.

Если вы хотите создать макрос с аргументами, имя которого совпадает с именем уже существующей команды, то надо воспользоваться командой \renewcommand с необязательным аргументом. Место постановки и значение этого необязательного аргумента, а также правила употребления символов #1, #2 и т. д. при этом будут такие же, как для команды \newcommand.

Приведем еще один пример практически полезного макроса с аргументами. При написании этой книги автор широко пользовался ссылками на страницу, автоматически генерируемыми с помощью команды \pageref. Например, если какое-то место в тексте было помечено с помощью команды \label{units}, то ссылка на соответствующую страницу выглядела бы так:

Как мы уже отмечали на с. ~\pageref{units}, ...

После первого десятка таких ссылок возникает желание сократить число нажатий на клавиши. В результате в преамбуле появилась строка

\newcommand{\cstr}[1]{стр. ~\pageref{#1}}

и ссылки на страницы стало возможно оформлять так:

Как мы уже отмечали
на \cstr{units}, ...

Не говоря уж об экономии времени на набор команды \pageref, исходный текст, в котором сократилось количество латинских букв, стал более обозримым²¹.

²¹ Автор вначале не знал, что надо писать «с.», а не «стр.», но мгновенно исправился, всего лишь удалив две буквы из определения команды \cstr!

1.3. Новые возможности $\text{\LaTeX}'e 2\epsilon$

В $\text{\LaTeX}'e 2\epsilon$ предусмотрен целый ряд дополнительных способов создания новых команд. Мы рассмотрим только один из них (остальные представляют интерес только для \TeX ников, пишущих новые стилевые пакеты). Именно, в $\text{\LaTeX}'e 2\epsilon$ существуют команды $\backslash\text{newcommand}$ и $\backslash\text{gnewcommand}$ — варианты «со звездочкой» (см. с. 25) знакомых нам \newcommand и \gnewcommand . Они работают точно так же, как их тезки без звездочек, со следующим отличием: если команда с аргументами определена с помощью \newcommand* или \gnewcommand* , то в ее аргументе *не может содержаться пустая строка или команда \par*. Если ваша команда определена с помощью \newcommand или \pnewcommand без звездочки (в $\text{\LaTeX}'e 2.09$ только так и можно), то никаких ограничений на этот счет нет.

На первый взгляд это кажется не усовершенствованием, а нелепостью: зачем вводить новые запреты?! Дело тут вот в чем. В подавляющем большинстве случаев команды с аргументами, которые вы определяете, все равно не будут предусматривать подстановку вместо аргумента фрагмента текста, содержащего пустую строку или \par , так что в нормальных условиях этот запрет ни на чём не скажется (кстати, во всех примерах, приведенных в этой книге к настоящему моменту, можно совершенно безболезненно заменить \newcommand и \gnewcommand на их варианты «со звездочкой»). А вот диагностика ошибок при наличии этого запрета облегчается. В самом деле, представим себе, что вы забыли набрать закрывающую фигурную скобку в аргументе команды (весьма распространенная ошибка!), как в следующем примере (забыта фигурная скобка в самой первой формуле; команду \symbol мы определили в учебных целях на с. 216):

Символ Лежандра $\text{\symbol}{a[p]}$, где a не делится на p , равен по определению 1 , если a является квадратичным вычетом по модулю p , и -1 в противном случае.

Вопрос о том, как зависит $\text{\symbol}{a[p]}$ от p при фиксированном a , является весьма важным и трудным.
Великий немецкий математик К.-Ф. Гаусс...

Если пустые строки в аргументе команды \symbol не запрещены (так оно и будет, если определять \symbol с помощью \newcommand без звездочки, как у нас на с. 216), то \TeX будет терпеливо ждать, когда ему встретится закрывающая фигурная скобка, парная к первой из открывающих фигурных скобок в первой строке, а пока таковой нет — рассматривать весь читаемый им текст как составную часть первого аргу-

мента команды `\символ`. В конце концов ТЕХ либо доловит вас, что он прочел уже весь файл, а закрывающей фигурной скобки так и не нашел, либо, что хуже, прервет работу, объявив, что ему не хватило памяти.

Если же мы определим `\символ` с помощью `\newcommand*`, вот так,

```
\newcommand*{\символ}[2]{\left(\frac{#1}{#2}\right)}
```

то ошибка не пойдет дальше текущего абзаца: как только ТЕХ увидит пустую строку среди текста, рассматриваемого им как аргумент команды, он тут же прервет дальнейшее чтение и выдаст следующее стандартное сообщение об ошибке:

Runaway argument?

{`a{p}`\$, где `$a$` не делится на `$p$`, равен п\ETC.

! Paragraph ended before `\символ` was complete.

<to be read again>

`\par`

1.8

?

Нажав пару раз на «ввод», мы сможем благополучно продолжить обработку текста. В первом абзаце пропадет все, начиная со слов «Символ Лежандра», зато второй и последующие абзацы будут обработаны нормально (пока ТЕХ не наткнется на очередную ошибку ...).

`\newcommand*` и `\renewcommand*` обладают еще одним преимуществом перед своими вариантами без звездочек: при их использовании происходит (неменьшая) экономия памяти.

Закончим этот раздел следующей рекомендацией: при пользовании ИАТЕХ'ом 2ε определяйте и переопределяйте команды при помощи `\newcommand*` и `\renewcommand*`. Варианты без звездочек используйте только тогда, когда вы действительно намерены подставлять в аргумент своего макроса текст, состоящий из нескольких абзацев.

2. Счетчики

В этом разделе мы научимся самостоятельно организовывать автоматическую нумерацию, подобно тому, как ИАТЕХ автоматически нумерует главы, страницы, формулы и т. д. Для этого нам надо познакомиться с понятием счетчика. Счетчик — это специальная переменная, значение которой является целым числом. ИАТЕХ предоставляет возможность присваивать счетчику различные значения и изменять их, выводить значение счетчика на печать, а также организовывать с помощью счетчиков

автоматическую генерацию ссылок. Рассмотрим последовательно, как все это делается.

2.1. Создание счетчиков и простейшие операции с ними

Каждый *L^AT_EX*'овский счетчик имеет свое имя — последовательность букв (без знака \). Прописные и строчные буквы в именах счетчиков различаются; если русские буквы можно употреблять в именах команд, то их можно употреблять и в именах счетчиков. Чтобы можно было работать со счетчиком, надо его создать командой `\newcounter`, имеющей один обязательный аргумент — имя, которое вы придумали для счетчика. Например, команда

```
\newcounter{abcd}
```

создает новый счетчик с именем `abcd`. Если имя, которое вы придумали для счетчика, уже занято (так может случиться, даже если команда `\newcounter` в вашем тексте всего одна: некоторые счетчики в *L^AT_EX*'е уже определены в момент начала обработки вашего текста), то *L^AT_EX* создавать счетчик с таким именем откажется и выдаст сообщение об ошибке.

В отличие от многих других *L^AT_EX*'овских команд, команда для создания нового счетчика является «глобальной»: даже если она давалась внутри группы, *L^AT_EX* не забудет о существовании определенного ею счетчика и после выхода из этой группы. Это отличает `\newcounter` от `\newcommand` и `\renewcommand`.

Что же можно делать со счетчиком? Во-первых, можно менять его численное значение (на программистском жаргоне: «присваивать счетчику различные значения»). При создании счетчика его значение устанавливается в 0; чтобы установить какое-то другое значение, используется команда `\setcounter`, имеющая два обязательных аргумента: первый — имя счетчика, второй — значение, которое счетчику присваивается. Если, например, написать

```
\setcounter{abcd}{1995}
```

то после того, как *T_EX* прочтет эту команду, значение счетчика `abcd` установится равным 1995. Значение, которое присваивается счетчику, может быть и отрицательным, но обязано быть целым.

Другой способ изменить значение счетчика — это прибавить к нему какое-то целое число. Для этого используется команда `\addtocounter`. Эта команда также имеет два обязательных аргумента: первый — имя счетчика, второй — число, которое прибавляется к счетчику. Например, после выполнения команд

```
\setcounter{abcd}{100}
\addtocounter{abcd}{-27}
```

значением счетчика `abcd` будет число 73.

Команды, изменяющие значение счетчика, также являются «глобальными»: если с их помощью внутри группы значение счетчика было изменено, то по выходе из группы его прежнее значение не восстановится.

Главная задача L^AT_EX'а — не просто присваивать значения каким-то переменным, а верстать тексты, так что перейдем к самому главному: как выдать значение счетчика на печать. Самый распространенный случай — печать значения счетчика обычными («арабскими») цифрами. Для этого используется команда `\arabic`:

Шел по улице отряд — 40 мальчи-
ков подряд.

```
\setcounter{abcd}{40}
Шел по улице отряд ---
\arabic{abcd}
мальчиков подряд.
```

Значение счетчика печатается текущим шрифтом: если значение счетчика равно, скажем, 1995, то, встретив команду `\arabic{abcd}`, Т_EX отреагирует так же, как если бы на ее месте в исходном тексте было написано 1995.

Чтобы напечатать значение счетчика римскими цифрами, надо воспользоваться командой `\Roman` (если мы хотим, чтобы римские цифры записывались прописными латинскими буквами) или `\roman` (чтобы записать римскую цифру строчными латинскими буквами):

Людовика XIV звали «Король-
Солнце».

```
\setcounter{abcd}{14}
Людовика`\Roman{abcd} звали
\лк Король-Солнце\пк.
```

Естественно, при печати значения счетчика римскими цифрами это значение должно быть положительным числом.

Можно, наконец, напечатать букву латинского алфавита, порядковый номер которой равен значению счетчика. Для этого используются команды `\alph` (для печати строчной буквы) и `\Alph` (для печати прописной буквы):

Наконец я узнаю, какая буква стоит в латинском алфавите на седьмом месте! Вот она: g.

```
\setcounter{abcd}{7}
Наконец я узнаю, какая буква
стоит в латинском алфавите
на седьмом месте!
Вот она: `"\alph{abcd}"
```

Если значение счетчика при пользовании этими командами превышает количество букв в латинском алфавите, то \LaTeX выдает сообщение об ошибке.

В \LaTeX 'е отсутствует команда, позволяющая напечатать *русскую* букву с номером, равным значению счетчика. Средствами \TeX 'а такую команду нетрудно создать, что и сделано, например, в «русифицирующем стиле», использованном при подготовке этой книги.

Наконец, можно напечатать один из девяти символов, используемых иногда в англоязычных странах для обозначения последовательных сносок (вместо цифр). Для этого используется команда $\backslash fnsymbol$, применять которую можно только внутри формул:

Для сносок в Англии применяют такие символы: *, †, ‡, а дальше попробуйте сами.

$\backslash setcounter{abcd}{0}$

Для сносок в Англии

применяют такие символы:

$\$ \backslash addtocounter{abcd}{1} \backslash fnsymbol{abcd} \$,$
 $\$ \backslash addtocounter{abcd}{1} \backslash fnsymbol{abcd} \$,$
 $\$ \backslash addtocounter{abcd}{1} \backslash fnsymbol{abcd} \$,$
 а дальше попробуйте сами.

Обратите внимание, как три идентичных фрагмента исходного текста дали на печати три разных символа.

Для полноты картины скажем об еще одной менее употребительной команде, связанной со счетчиками. Именно, в командах для изменения значений счетчиков можно вместо явного указания чисел использовать значения других счетчиков, для чего употребляется команда $\backslash value$. Например, в результате выполнения команд

```
\newcounter{efgh}
\setcounter{abcd}{10}
\setcounter{efgh}{100}
\addtocounter{efgh}{-\value{abcd}}
```

значение счетчика *efgh* станет равно 90. Можно даже писать, например,

```
\setcounter{efgh}{1000}
\tolerance=\value{efgh}
```

но большого смысла в таких трюках, как правило, нет.

Давайте теперь применим наши познания о счетчиках для дела. На с. 214 мы обещали вам так усовершенствовать макрос *\z*, начинающий новый абзац и печатающий жирным шрифтом слово «Задача», чтобы он еще и автоматически нумеровал эти задачи, так, что можно было бы просто писать в исходном тексте

```
\z Найти сумму...
\z Решить уравнение...
\z Поезд вышел из пункта А...
```

и при этом знать, что номера L^AT_EX проставит сам. Теперь мы в состоянии решить эту проблему. Во-первых, для этого надо создать счетчик, значение которого в каждый момент будет равно номеру последней обработанной задачи; во-вторых, в определении команды \z надо предусмотреть, чтобы всякий раз значение этого счетчика увеличивалось на единицу, а затем печаталось в качестве номера задачи. В качестве имени счетчика выберем бесхитростное **задача**:

```
\newcounter{задача}
```

(напомним, что при выполнении этой команды счетчику **задача** будет присвоено значение 0). Теперь модифицируем определение макроса \z так:

```
\newcommand{\z}{\par\addtocounter{задача}{1}%
{\bf Задача \arabic{задача}.}}
```

Напомним, что команда \par означает «завершить предыдущий абзац, если он еще не был завершен»; без нее можно обойтись, если мы будем ставить команду \z только после пустой строки. Знак процента мы поставили, чтобы убрать лишний пробел, порождаемый концом строки. Теперь при первом исполнении команды \z значение счетчика **задача** станет равно 1 и будет напечатано «Задача 1.», при втором исполнении этой команды значение счетчика станет равно уже 2 и напечатается «Задача 2.» ... и т. д., что нам и нужно!

2.2. Отношение подчинения между счетчиками

Команда \z, как мы ее определили в предыдущем разделе, нумерует задачи автоматически, но при этом нумерация получается «сплошной». Часто, однако, требуется, чтобы в каждом разделе документа нумерация задач начиналась заново, так что шестая задача в разделе с номером 3 была озаглавлена **Задача 3.6**, а первая задача в разделе с номером 4 — **Задача 4.1**. Сейчас мы узнаем, как этого добиться.

Выше мы уже упоминали, что к моменту начала обработки L^AT_EX'ом нашего текста некоторые счетчики уже определены. В частности, это счетчики, содержащие номера текущих разделов документа. Их имена совпадают с именами команд, генерирующих эти разделы: **chapter** (если стилем предусмотрено разбиение на главы), **section**, **subsection**

и т. д. При каждом исполнении, например, команды `\section` значение счетчика `section` увеличивается на 1, и значение этого счетчика в каждый момент равно номеру текущего раздела. Поэтому, если в определении команды `\z` написать

```
\arabic{section}.\arabic{задача}.
```

то перед номером задачи будет печататься номер текущего раздела и точка.

Но как же, все-таки, сделать, чтобы в каждом разделе нумерация задач начиналась заново? Можно, конечно, после каждой команды `\section` присваивать счетчику `задача` значение 0 с помощью команды `\setcounter`, но это некрасиво и ненадежно (а вдруг забудем?). Вместо этого следовало бы сразу определить счетчик `задача` таким образом:

```
\newcounter{задача}[section]
```

При таком определении счетчик `задача` будет, как говорят, *подчинен* счетчику `section`: всякий раз, когда значение счетчика `section` увеличивается на единицу командой `\section`, значение счетчика `задача` будет устанавливаться в нуль, и тем самым счет задач будет в каждом разделе начинаться заново. Стало быть, если считать, что счетчик `задача` определен именно так, как сказано выше, то очередной раз исправленное определение команды `\z` выглядит так:

```
\newcommand{\z}{\par\addtocounter{задача}{1}%
{\bf Задача } \arabic{section}.\arabic{задача}.}
```

При этом нумерация задач будет начинаться заново в каждом разделе, и вторая задача третьего раздела будет иметь номер 3.2.

Теперь сообщим точные правила создания счетчиков, подчиненных другому счетчику. Они просты: команда `\newcounter` может принимать один необязательный аргумент (*после обязательного*) — имя того счетчика, которому будет подчинен определяемый нами счетчик. Разумеется, в момент выполнения команды `\newcounter` с необязательным аргументом счетчик, имя которого дается в квадратных скобках, должен уже существовать.

Ко всему сказанному требуется одно важное уточнение: мы еще толком не объяснили, в каких случаях значение подчиненного счетчика устанавливается в нуль. В самом деле, пусть счетчик слуга подчинен счетчику `хозяин`; тогда команда

```
\addtocounter{хозяин}{1}
```

никоим образом не повлияет на значение подчиненного счетчика слуга: изменение значений счетчика влияет на значения подчиненных ему счетчиков только в том случае, если значение подчиняющего счетчика изменялось с помощью специальных команд. Таких команд всего две, из них чаще всего используется `\refstepcounter`: она увеличивает на единицу значение счетчика, имя которого является ее аргументом, а значения всех счетчиков, подчиненных счетчику — ее аргументу, устанавливает в нуль. Если, например, в нашем тексте определены два счетчика так:

```
\newcounter{хозяин}
\newcounter{слуга}[хозяин]
```

то после выполнения команд

```
\setcounter{хозяин}{10}
\setcounter{слуга}{10}
```

значения обоих счетчиков станут равны 10, после выполнения команды

```
\addtocounter{хозяин}{1}
```

значение счетчика **хозяин** станет равно 11 и значение счетчика слуга не изменится, а вот после выполнения команды

```
\refstepcounter{хозяин}
```

значение счетчика **хозяин** станет равно 12, в то время как значение счетчика слуга станет равно нулю.

Наряду с `\refstepcounter` существует еще одна команда, изменяющая значение счетчика таким образом, что значения всех подчиненных ему счетчиков устанавливаются в нуль. Эта команда называется `\stepcounter`; она также увеличивает на единицу значение счетчика, имя которого является ее аргументом, и при этом обнуляет все подчиненные ему счетчики, но она непригодна для организации автоматических ссылок (см. следующий раздел), вследствие чего область ее применения более ограничена.

Хороший пример использования подчиненных счетчиков дают стандартные стили L^AT_EX'a. Например, если основным стилем является `book`, то перед началом обработки текста выполняются следующие команды:

```
\newcounter {part}
\newcounter {chapter}
\newcounter {section}[chapter]
\newcounter {subsection}[section]
\newcounter {subsubsection}[subsection]
\newcounter {paragraph}[subsubsection]
\newcounter { subparagraph}[paragraph]
```

Стало быть, нумерация глав не зависит от нумерации частей (если третья часть книги завершается десятой главой, то четвертая часть начинается с одиннадцатой главы), а нумерация разделов уже начинается заново в каждой главе.

2.3. Организация автоматических ссылок

Вернемся еще один, теперь уже последний, раз к нашей команде `\z`. Раз она автоматически нумерует задачи, то неплохо было бы, если бы пронумерованные ею задачи можно было метить командой `\label` и ссылаться на эти метки командой `\ref` (проблема именно в ней, поскольку команда `\pageref`, дающая номер страницы, сработает в любом случае). Если коротко, то решение этой проблемы таково: увеличивать на единицу значение счетчика `задача` надо не с помощью команды `\addtocounter`, которой мы пользовались до сих пор, а с помощью команды `\refstepcounter`, о которой уже шла речь по другому поводу в предыдущем разделе. Если мы определим команду `\z` так:

```
\newcommand{\z}{\par\refstepcounter{задача}%
{\bf Задача } \arabic{section}. \arabic{задача}.}
```

то после этого можно будет написать, например, так:

```
\z Решить уравнение...
\z Доказать... \label{prove}
\z Найти сумму...
```

Если теперь в другом месте текста мы сошлемся на помеченную задачу так:

В задаче `\ref{prove}` предлагалось доказать...

то будет печататься ее номер (тот самый, который L^AT_EX автоматически ей присвоил). Впрочем, с такими автоматическими ссылками не все будет благополучно: если помеченная нами задача была второй по счету в разделе номер 3, то называться она будет **Задача 3.2**, а вот ссылка на нее, сгенерированная командой `\ref`, будет выглядеть просто

В задаче 2 предлагалось доказать ...

в то время как хотелось бы автоматически получить «В задаче 3.2». Иными словами, надо изменить текст, генерируемый командой `\ref`. Чтобы узнать, как этого добиться, нам придется познакомиться с еще одной L^AT_EX'овской конструкцией, связанной со счетчиками.

Мы уже знаем, что значение L^AT_EX'овского счетчика можно вывести на печать командами `\arabic`, `\roman` и т. п. Однако, кроме этого,

с каждым счетчиком связана индивидуальная команда, определяющая, в какой форме его значение будет выводиться на печать, и именно в соответствии с этой командой печатается ссылка, сгенерированная с помощью `\label` и `\ref`. Имя этой команды получается, если поставить `the` перед именем счетчика. Например, команда для вывода на печать номера раздела называется `\thesection`, для вывода на печать номера главы — `\thechapter`, команды для вывода на печать определенных нами счетчиков слуга и хозяин — `\thesluga` и `\thexozain` соответственно. При создании счетчика автоматически определяется и соответствующая `the`-команда. Например, при создании счетчика `abcd` автоматически определяется команда `\theabcd` таким образом:

```
\newcommand{\theabcd}{\arabic{abcd}}
```

В дальнейшем эту команду можно переопределять:

Людовика XIV звали «Король-Солнце».

```
\setcounter{abcd}{14}
\renewcommand{\theabcd}%
{\Roman{abcd}}
Людовика^{\theabcd{}} звали
\лк Король-Солнце\пк.
```

Мы же, чтобы при ссылках перед номером задачи печатался номер раздела, в котором находится эта задача, и точка, переопределим команду `\the задача` так:

```
\renewcommand{\the задача}{\thesection.\arabic{задача}}
```

Если включить эту команду в преамбулу документа, то ссылки на сгенерированные нашей командой `\z` номера задач будут выглядеть должным образом.

В нашем переопределении команды `\the задача` мы воспользовались командой `\thesection`, чтобы наши макросы правильно работали с любым основным стилем документа. Дело в том, что при разумном оформлении номер раздела, предшествующий при ссылке номеру задачи, должен печататься таким же образом, как и номер раздела при ссылке на раздел, а это в разных стилях делается по-разному: в стиле `article`, например, `\thesection` — это то же самое, что и `\arabic{section}` (иными словами, ссылка на раздел, сгенерированная командой `\ref`, печатает просто номер раздела), а в стиле `report` команда `\ref` при печати ссылки на раздел печатает не номер раздела, а номер главы, точку и номер раздела. Поскольку мы написали `\thesection`, все эти тонкости будут учтены автоматически.

Приведем еще один (игрушечный) пример использования счетчиков в макроопределениях, отчасти чтобы еще раз продемонстрировать

применение подчиненных счетчиков, а отчасти чтобы убедить читателя, что не боги горшки обжигают. Именно, давайте разработаем свои собственные команды для создания разделов документа, не полагаясь на `\chapter`, `\section` и т. п. из стандартных L^AT_EX'овских стилей. Поскольку пример игрушечный, предположим, что заголовки глав и разделов будут укладываться в одну строку и не будем заботиться о том, насколько удачным выйдет оформление заголовков. Итак, приступим. Пусть наш документ делится на главы, которые, в свою очередь, делятся на разделы. Каждую главу будем начинать с новой страницы, перед каждым разделом оставлять 1 см (если, конечно, раздел не начинает новой страницы). Наконец, предусмотрим, чтобы главы и разделы автоматически нумеровались с возможностью создания автоматических ссылок на эти номера. Команды для создания главы и раздела назовем `\глава` и `\раздел` соответственно; это будут команды, требующие одного аргумента — названия главы или раздела.

Для того чтобы номера глав и разделов генерировались автоматически, нам необходимо создать счетчики, содержащие эти номера. Пусть счетчик с номером главы называется `глава`, а счетчик с номером раздела — `раздел` (имена счетчиков могут совпадать с именами команд). Имея в виду, что нумерация разделов в каждой главе будет начинаться заново, напишем в преамбуле так:

```
\newcounter{глава}
\newcounter{раздел}[глава]
```

Теперь определим команду `\глава`:

```
\newcommand{\глава}[1]{\clearpage % с новой страницы
\vspace*{4cm}%
% оставить место сверху
\refstepcounter{глава}%
% новый номер главы
{\LARGE\bf %
% шрифт для заголовка
Глава \thegлава. #1% заголовок
\par %
% кончить заголовок
\vspace{5mm plus 1mm minus .5mm}%
% Промежуток между
% заголовком и текстом
}%
% завершить группу, внутри которой менялся шрифт
}%
% конец макроопределения
```

Поскольку номер главы устанавливается командой `\refstepcounter`, при этом будет начата заново и нумерация разделов, а на номера глав можно будет делать автоматические ссылки с помощью `\label` и `\ref`. Обратите также внимание на команду `\thegлава`. Мы воспользовались ею, поскольку не хотим на этом этапе предрешать, в каком виде номер

главы будет представлен в заголовке: в виде арабской цифры, римской цифры или еще как-нибудь. Если в дальнейшем нам захочется изменить вид этого представления, то не придется лезть, с риском ошибиться, в длинное определение команды `\глава`, а будет достаточно переопределить команду `\theглава`. Заметьте, что промежуток между заголовком и текстом мы сделали *растяжимым*, чтобы помочь TeX'у найти правильное разбиение на страницы (см. с. 122).

Определение команды `\раздел` можно дать, например, так:

```
\newcommand{\раздел}[1]{\par % завершить предыдущий абзац
\pagebreak[2]\vspace{1cm plus 3mm minus .5mm} % см. ниже
\refstepcounter{раздел}% новый номер раздела
{\Large\bf % шрифт для заголовка
\therаздел{} #1% заголовок
\par % кончить заголовок
} % завершить группу, внутри которой менялся шрифт
\nopagebreak % чтобы не оторвать текст от
               % заголовка
\vspace{2mm plus 1mm} % Промежуток между заголовком
                      % и текстом
} % конец макроопределения
```

Пояснений тут требует команда `\pagebreak[2]`. Мы включили ее в макроопределение, чтобы поменьше разделов начиналось внизу страницы. В самом деле, команда `\pagebreak[2]` предлагает TeX'у начать в этом месте новую страницу (см. раздел III.8.3); если так и будет сделано, то дополнительный вертикальный промежуток, созданный командой `\vspace`, пропадет, и заголовок раздела начнется с самого верха новой страницы; если же разрыва страницы все-таки не произойдет, то перед заголовком раздела будет вертикальный промежуток величиной 1 см (обладающий указанными в макроопределении *растяжимостью* и *сжимаемостью*).

Нам осталось только задать вид, в котором будут представляться на печати номера глав и разделов. Иными словами, надо переопределить должным образом команды `\theглава` и `\therаздел` (в момент создания счетчиков они, как мы помним, были автоматически определены таким образом, что `\theглава` и `\therаздел` — просто номер главы и раздела соответственно, набранный арабскими цифрами). Предположим, мы решили, что номера глав будут печататься римскими цифрами, а номер второго раздела четвертой главы будет иметь вид IV-2. Тогда требуемые переопределения таковы:

```
\renewcommand{\theглава}{\Roman{глава}}
```

```
\renewcommand{\therazdel}{\Roman{глава}--\arabic{раздел}}
```

Еще одно замечание: точки после номера главы мы включили в определение команды \глава, а не \thegлава, чтобы можно было пользоваться автоматическими ссылками: если бы \thegлава определялось как

`\Roman{глава}.`

то исходный текст

в главе `\ref{метка}` мы пишем...

дал бы на печати

в главе IV. мы пишем

что нелепо.

По сравнению с макроопределениями, реально используемыми в стандартных стилях *Л^AT_EX*'а, мы в нашем игрушечном наборе макросов многое не предусмотрели: не позаботились ни о колонтитулах, ни об автоматически генерируемом оглавлении, внешний вид заголовков оставляет желать лучшего, и т. д. Тем не менее один из основных принципов у нас присутствует: нумерация действительно организуется с помощью подчиненных друг другу счетчиков и команды `\refstepcounter`.

У внимательного читателя может возникнуть вопрос, каким образом команда `\label` узнаёт, какого вида ссылку ей генерировать. В самом деле, допустим, что исходный текст имеет вид

```
\chapter{Что-то}\label{a}
...
\section{Кое-что}\label{b}
...
\begin{figure}
\caption{Еще кое-что}\label{c}
...
\end{figure}
```

и глава оказалась по номеру третьей, помеченный раздел — вторым в этой главе, а подписанная плавающая иллюстрация — пятой в своем разделе. Откуда *Л^AT_EX* знает, что команда `\ref{a}` должна сгенерировать просто цифру 3, команда `\ref{b}` — текст 3.2, а команда `\ref{c}` — текст 3.2.5? Ответ на этот вопрос таков: команда `\label` генерирует (т. е. записывает в aux-файл) ссылку в соответствии с видом того счетчика, который последним подвергался операции `\refstepcounter`. Поэтому в данном примере команда `\label{a}` генерирует ссылку на счетчик `chapter`, `\label{b}` — на счетчик `section`, а `\label{c}` — на счетчик `figure`, отвечающий за нумерацию плавающих иллюстраций.

2.4. Счетчики, которые определять не надо

Мы уже мельком упоминали, что при начале работы L^AT_EX'a некоторые счетчики определены сразу. Например, это те счетчики, которые перечислены на с. 225. Кроме того, заранее определен очень важный счетчик `page`, отвечающий за нумерацию страниц, а также счетчик `footnote`, ответственный за нумерацию сносок. Нумерацией плавающих иллюстраций и таблиц занимаются счетчики, называемые попросту `figure` и `table`. В разд. IX.2 перечислены все эти заранее определенные счетчики и указано, каким счетчикам они подчинены (подчиненность иногда зависит от стиля документа).

Для каждого из этих счетчиков вы имеете возможность переопределить соответствующую `\the-`команду и тем самым изменить стиль оформления документа. Например, вы можете сделать так, чтобы главы нумеровались римскими цифрами:

```
\renewcommand{\thechapter}{\Roman{chapter}}
```

Если вы хотите, чтобы сноски нумеровались не цифрами, а латинскими буквами, то можно в преамбуле написать:

```
\renewcommand{\thefootnote}{\alph{footnote}}
```

2.5. Модификация оформления перечней

Хороший пример переопределения команд доставляют перечни. В свое время (разд. III.9) мы обещали рассказать о том, как менять оформление перечней, задаваемых окружениями `itemize` и `enumerate`; сейчас мы, наконец, можем это сделать.

Начнем с `itemize`. Чтобы поменять значки, которыми помечаются элементы перечня, надо переопределить команду `\labelitemi`. Если, например, мы хотим, чтобы элементы перечня отмечались не черными кружками, а такими галочками \checkmark , то достаточно написать в преамбуле

```
\renewcommand{\labelitemi}{$\surd$}
```

(команду `\surd` см. в таблице на с. 50). Если окружение `itemize` расположено внутри другого окружения `itemize`, как в примере на с. 128, то значки для пометки элементов перечня будут уже, вообще говоря, другими: их вид задается командой `\labelitemii`; вид значков для пометок элементов `itemize` на третьем и четвертом уровнях вложенности задается командами `\labelitemiii` и `\labelitemiv` соответственно; их также можно переопределять.

Правильнее было бы определять заголовки для `itemize` чуть хитрее. Например, наше определение `\labelitemi` лучше дать так:

```
\renewcommand{\labelitemi}{\$\mathsurround=0pt \surd\$}
```

Если дать определение именно так, то вокруг галочки не появится дополнительный пробел даже в случае, если вы в какой-то момент решите установить ненулевое значение параметра `\mathsurround`. Так как формула образует группу, в дальнейшем предыдущее значение `\mathsurround` восстановится. Полезно иметь в виду этот прием, если вы пользуетесь математическими символами в качестве типографских значков.

Теперь рассмотрим окружение `enumerate`. Коль скоро оно автоматически нумерует элементы перечня, можно предположить, что это окружение связано с L^AT_EX'овскими счетчиками. Так оно на самом деле и есть: это окружение использует счетчик `enitm`. Если одно `enumerate` вложено в другое, то используются счетчики `enumii`, `enumiii` и `enumiv` для нумерации элементов перечня на втором, третьем и четвертом уровнях вложенности соответственно. С другой стороны, сами значки, помечающие элементы перечня, порождаются командами `\labelenumi`, `\labelenumii`, `\labelenumiii` и `\labelenumiv` соответственно — в зависимости от уровня вложенности. Например, в стандартных L^AT_EX'овских стилях команда `\labelenumi` определена так:

```
\newcommand{\labelenumi}{\theenumi.}
```

в то время как `the`-команда, определяющая представление счетчика `enitm` на печати, определена просто как

```
\newcommand{\theenumi}{\arabic{enumi}}
```

Стало быть, если мы не меняем стандартного стиля, то элементы перечня `enumerate` (не вложенного в другой `enumerate`) будут нумероваться цифрами с точкой. Если же мы хотим, скажем, чтобы после цифры шла не точка, а скобка (как в нашей книге), то можно в преамбуле написать

```
\renewcommand{\labelenumi}{\theenumi)}
```

Если же мы к тому же хотим, чтобы элементы перечня нумеровались римскими цифрами, то можно написать еще и так:

```
\renewcommand{\theenumi}{\Roman{enumi}}
```

Аналогичным образом можно менять оформление нумерованных перечней на других уровнях вложенности.

Если вы хотите изменить оформление перечня более серьезным образом (например, установить другую величину полей, или сделать так, чтобы значки, помечающие элементы перечня, были выровнены по левому, а не по правому краю), то вам придется подождать до гл. IX.

3. Параметры со значением длины

Наряду с со счетчиками — переменными с целочисленными значениями, при создании собственных макроопределений возникает нужда и в переменных, значениями которых являются длины. Например, в предыдущем разделе мы, разрабатывая команду `\раздел`, в явном виде задали промежуток между заголовком раздела и остальным текстом. Если этот промежуток нам почему-либо захочется изменить, то придется снова залезать в определение команды `\раздел`. Было бы удобнее, если бы в нашем распоряжении был параметр под названием, скажем, `\отступ`, так что можно было бы в определении команды `\раздел` написать

```
\vspace{\отступ}
```

и потом отдельно написать, допустим,

```
\отступ=2mm
```

(или присвоить параметру `\отступ` другое значение). Правда, в богатом наборе \TeX'овских и \LaTeX'овских параметров требуемого нам параметра `\отступ` нет. Но это не страшно, поскольку его можно создать. Для этого используется команда `\newlength`:

```
\newlength{\отступ}
```

После того, как вы, допустим, в преамбуле, дали эту команду, будет определен новый параметр со значением длины; его можно будет обычным образом использовать в аргументах команд наподобие `\vspace` и ему можно будет обычным образом присваивать значения.

Теперь — точные правила. Команда `\newlength` имеет один обязательный аргумент — имя команды, обозначающей определяемый вами параметр. Это имя должно подчиняться обычным правилам для \TeX'овских команд (backslash, после которого следует либо одна не-буква, либо последовательность букв). Если это имя уже занято, \LaTeX выдаст сообщение об ошибке. Определение нового параметра, совершающее коммандой `\newlength`, является «глобальным»: даже если эта команда была дана внутри группы, \TeX будет помнить о существовании этого параметра и по выходе из группы. По этой, в частности, причине разумное место для комманды `\newlength` — преамбула.

Определенный нами параметр со значением длины приобретает такой же статус, как уже существующие \TeX'овские и \LaTeX'овские параметры, такие, как `\parindent`, `\textwidth`, и т. п. Рассмотрим, что можно делать с этими параметрами.

Во-первых, параметрам со значением длины можно присваивать значения. Делается это точно так же, как это объяснялось в разд. I.2.6 на

примере параметра `\parindent`: для присваивания значения надо написать имя параметра, знак равенства, и после знака равенства — величину присваиваемой длины. Пробелы после указания единицы длины $\text{\TeX}'$ ом игнорируются (скорее всего, вы будете присваивать значения параметрам в преамбуле документа или между абзацами, где лишние пробелы никого не волнуют). Длина должна быть выражена в единицах, воспринимаемых $\text{\TeX}'$ ом (см. их список в разд. I.2.10). Даже если вы присваиваете нулевую длину, какая-то единица длины должна быть явно указана (например, `Opt`). Кроме того, можно воспользоваться $\text{\LaTeX}'$ овской командой `\setlength`, имеющей два обязательных аргумента: первый — имя параметра, второй — значение длины, присваиваемое этому параметру. Таким образом, команды

`\parindent=1.5em`

и

`\setlength{\parindent}{1.5em}`

равносильны. Наконец, отметим, что, как мы уже неоднократно отмечали, присваивания, сделанные внутри группы, забываются по выходе из этой группы.

В предыдущем абзаце мы немного скучавили, умолчав об одной возможной неприятности. Дело в том, что, если после команды присваивания, не использующей `\setlength`, следует (пусть даже после пробела) слово `plus` или `minus`, то \TeX , скорее всего, выдаст вам сообщение об ошибке, поскольку решит, что длина должна иметь, помимо «естественногоразмера», еще и `plus-` или `minus-`компоненту (см. с. 122; ниже мы поговорим подробнее о такой возможности). Если вы пишете текст на русском языке, вероятность такого стечения обстоятельств ничтожна. Тем не менее, забывать о такой опасности не следует, и особенно, если команда присваивания входит в макроопределение: вы же не знаете заранее, в какое место вставите свой макрос. Чтобы застраховаться от этой неприятности раз и навсегда, пользуйтесь `\setlength`, хоть при этом и придется нажать на большее число клавиш. Ср. также обсуждение команд `\hrule` и `\vrule` в разд. III.10.

Параметры со значением длины можно использовать всюду, где в аргументе $\text{\LaTeX}'$ овской команды требуется указать размер. Пусть, например, в преамбуле документа написано

`\newlength{\пример}`

Тогда посмотрите на следующий пример:

9	9	<code>\пример=10mm</code>
8	8	<code>9\hspace{\пример}9</code>
9	9	<code>{\пример=20mm}</code> <code>8\hspace{\пример}8}</code>
<code>9\hspace{\пример}9</code>		

Обратите внимание, что, если присвоение параметру нового значения происходило внутри группы, то по выходе из группы новое значение забывается, а прежнее — восстанавливается.

Параметры со значением длины можно указывать с коэффициентом — положительной или отрицательной десятичной дробью (можно использовать как десятичную точку, так и десятичную запятую). Например, если значение параметра `\пример` равно 10 мм, то команда `\hspace{2.71\пример}` сделает пробел длиной 27.1 мм.

Можно также прибавлять длину к значению параметра: если значение параметра `\abcd` равно x , то после выполнения команды

`\addtolength{\abcd}{y}`

где y — длина, значение параметра `\abcd` станет равно $x + y$. В качестве y в этой команде может использоваться как явно указанная длина (например, `1.2in`), так и параметр со значением длины (возможно, с числовым коэффициентом). Наконец, L^AT_EX предоставляет еще одну полезную команду

`\settowidth{параметр}{текст}`

которая присваивает *параметру* значение, равное ширине *текста*. Вот пример:

СЛОВО	слово	<code>\settowidth{\пример}{\Large</code>
	слово	<code>слово }</code>
		<code>{\Large слово }слово</code>
<code>\hspace{\пример}слово</code>		

- ε В L^AT_EX'е определены также такие команды, как `\settoheight` и `\settodepth`, аналогичные `\settowidth`. Команда `\settoheight` присваивает *параметру* значение, равное максимальному расстоянию, на которое *текст* возвышается над строкой (точнее, над ее базисной линией — разд. VIII.1). `\settodepth` присваивает *параметру* значение, равное максимальному расстоянию, на которое *текст* опускается ниже базисной линии.

В разд. III.8.4 у нас шла речь о том, что некоторые используемые в $\text{\TeX}'е$ длины могут обладать растяжимостью или сжимаемостью. Параметрам, созданным с помощью команды `\newlength`, также можно присваивать значения, содержащие `plus-` и/или `minus-`компоненту. Если, например, мы хотим, чтобы параметр `\пример` имел естественный размер 2 см и при этом мог растягиваться на 4 мм и сжиматься на один пункт, то можно написать так:

```
\setlength{\пример}{2cm plus 4mm minus 1pt}
```

4. Создание новых окружений

4.1. Новые окружения: общий случай

Как мы уже имели возможность убедиться, для сокращения времени на написание длинных последовательностей команд удобно пользоваться макросами. В тех случаях, когда для достижения необходимого нам эффекта требуется сложная последовательность команд в начале и в конце какого-то текста, \LaTeX дает возможность оформить соответствующие макросы в виде нового окружения. Как это делается, разберем на примере.

Предположим, нам хочется взять в рамку абзац текста шириной 7 см. Один из возможных способов таков:

```
\begin{tabular}{|p{7cm}|}
```

```
\hline
```

Этот текст будет заключен в рамку. Как видите, окружение, предназначенное для верстки таблиц, можно использовать и для этих целей.\\"

```
\hline
```

```
\end{tabular}
```

что даст на печати вот что:

<p>Этот текст будет заключен в рамку. Как видите, окружение, предназначенное для верстки таблиц, можно использовать и для этих целей.</p>

Если таких рамок с текстом у вас много, то можно сократить число нажатий на клавиши, определив окружение с именем, скажем, `рамка`, так, чтоб можно было бы просто писать

```
\begin{рамка}
Этот текст будет ...
... этих целей.
\end{рамка}
```

Определяется это окружение так:

```
\newenvironment{рамка}{\begin{tabular}{|p{7cm}|}}
    \hline{\\\hline\end{tabular}}
```

В общем случае команда `\newenvironment` имеет такой формат:

```
\newenvironment{имя}{открывающие_команды}{закрывающие_команды}
```

Здесь `имя` — имя определяемого окружения, `открывающие_команды` — команды и/или текст, подставляемые вместо команды `\begin` с именем окружения, `закрывающие_команды` — команды и/или текст, подставляемые вместо команды `\end` с именем окружения.

Вместо окружения, определяемого с помощью `\newenvironment`, можно с тем же успехом создать два макроса: один — для `открывающих_команд`, другой — для `закрывающих`. Например, в нашем случае с рамкой можно было бы написать

```
\newcommand{\начать}{\begin{tabular}{|p{7cm}|}\hline}
\newcommand{\кончить}{\\\hline\end{tabular}}
```

и создавать рамки так:

```
\начать
Этот текст...
\кончить
```

Преимущество оформления такого рода конструкций в виде окружений состоит в том, что при этом легче контролировать ошибки: если вы напишете `\begin{рамка}` и при этом забудете написать соответствующую команду `\end{рамка}`, то `LATEX` выдаст сообщение об ошибке, в котором именно это вам и скажет; если же вы забудете команду `\кончить`, то сообщения об ошибке будут менее понятными. Кроме того, нелишне напомнить, что команды `\begin` и `\end`, ограничивающие окружение, ограничивают группу: все неглобальные определения и изменения параметров, происходящие внутри окружения, забываются по выходе из него.

Новые окружения можно определять так, чтобы они принимали аргументы. Пусть, например, в зависимости от обстоятельств нам нужны рамки разной ширины. Тогда разумно модифицировать определение окружения рамка таким образом, чтобы ширина текста в рамке передавалась ему как аргумент. Соответствующее определение будет выглядеть так:

```
\newenvironment{рамка}[1]{\begin{tabular}{|p{#1}|}
    \hline}
    \\\hline\end{tabular}}
```

После этого можно писать, например,

```
\begin{рамка}{6cm}
Текст...
\end{рамка}
```

или даже

```
\begin{рамка}{.85\textwidth}
Текст...
\end{рамка}
```

Общие правила такие. Чтобы создать окружение с аргументами, надо воспользоваться командой `\newenvironment` с необязательным аргументом. Этот необязательный аргумент ставится между первым и вторым обязательными; как и в случае с `\newcommand`, он означает количество аргументов, которые будет требовать окружение, и это количество не может превышать девяти; места, куда будут вставлены аргументы, по-прежнему обозначаются `#1`, ... `#9`, причем эти значки можно употреблять только в открывающих командах (т. е. во втором обязательном аргументе команды `\newenvironment`).

С помощью `\newenvironment` нельзя переопределить уже существующее окружение (если вы все же попробуете так сделать, L^AT_EX выдаст сообщение об ошибке). Если вам действительно необходимо такое переопределение, надо пользоваться командой `\renewenvironment`, работающей точно так же, как и `\newenvironment`, с тем различием, что в качестве первого аргумента ей можно передавать только имя уже существующего окружения.

- ε В L^AT_EX'е 2_ε определены также «варианты со звездочкой»: команды `\newenvironment*` и `\renewenvironment*`. Если окружение с аргументами определено с помощью одной из этих команд, то в его аргументе запрещены пустые строки или команды `\rag`.

4.2. Окружения типа «теорема»

Если вы пишете математический текст, то в этом тексте будет содержаться немалое количество теорем, лемм, определений и тому подобных вещей. Эти элементы математического текста желательно оформлять специальным образом. Например, формулировки теорем часто печатают, для ясности, другим шрифтом, само слово «теорема» также выделяют (третьим) шрифтом, и т. д. Чтобы задать такое оформление, в исходном тексте приходится написать довольно много \TeX овских команд, и лучше не повторять этот длинный набор команд много раз, а создать заменяющее его макроопределение, что, в свою очередь, может потребовать некоторого труда (чем-то подобным мы занимались в предыдущих разделах, когда разрабатывали команду $\text{\textbf{z}}$). Если же вы или редакция, с которой вы имеете дело, не слишком требовательны к деталям оформления, то соответствующие макросы (точнее говоря, новые окружения) легко создать из полуфабрикатов, предоставляемых нам для этих целей \LaTeX ом.

Окружения, используемые в \LaTeX е для оформления фрагментов текста типа «теорема», заранее не определены. Дело в том, что количество различных типов объектов наподобие теоремы, присутствующих в одном тексте, может быть достаточно велико (предложение, утверждение, лемма, определение, замечание ...), так что \LaTeX в целях экономии машинной памяти и исходя из того, что на все вкусы таких окружений не напасешься, определять их предоставляет вам. Как это делать, удобнее всего разобрать на примере.

Пусть в нашем тексте присутствуют «предложения». Давайте создадим окружение `predl` таким образом, чтобы можно было, например, писать

Предложение 1. Волга впадает в Каспийское море.

```
\begin{predl}
Волга впадает в Каспийское море.
\end{predl}
{\bf Доказательство.}
См. любую географическую карту.
```

Доказательство. См. любую географическую карту.

Для создания такого окружения используется команда `\newtheorem`:

```
\newtheorem{predl}{Предложение}
```

Как видите, команда `\newtheorem` имеет два обязательных аргумента: первый — название окружения, которое мы создаем, второй — заголовок нашей «теоремы».

Теперь обсудим, как работают окружения, созданные при помощи команды `\newtheorem` (будем называть их просто окружениями

типа «теорема»). Во-первых, как вы уже заметили, «формулировка» печатается курсивом, а заголовок — полужирным шрифтом. Во-вторых, абзац, идущий после нашего окружения, начинается с абзацным отступом, если после закрывающей окружение команды `\end` идет пустая строка, и без отступа в противном случае (так что в этом отношении окружения типа «теорема» ведут себя совершенно аналогично таким окружениям, как `quote`, `itemize` и т. п.). В-третьих, окружение типа «теорема» может иметь необязательный аргумент (как обычно, в квадратных скобках). Текст, стоящий в этих квадратных скобках, будет напечатан в скобках после заголовка «теоремы» и ее номера. Обычно это используется для указания ученого, чьим именем названа «теорема»:

Предложение 2 (Пифагор).
Пифагоровы штаны на все стороны равны.

```
\begin{predl}[Пифагор]
Пифагоровы штаны на
все стороны равны.
\end{predl}
```

- ε При пользовании *ЛМС-Л^AT_EX*овскими классами документов появляются дополнительные возможности влиять на оформление «теорем». См. приложение Е.

Вместе с окружением типа «теорема» автоматически создается и счетчик, хранящий его номер. Имя этого счетчика совпадает с именем окружения (так что в нашем примере счетчик называется `predl`); если мы хотим изменить представление на печати номеров нашей «теоремы», то можно обычным образом переопределить соответствующую `the-`команду. Например, если мы хотим, чтобы предложения нумеровались прописными латинскими буквами, надо в преамбуле написать:

```
\renewcommand{\thepredl}{\text{\textsf{Alph}}{\{predl\}}}
```

«Теоремы», определяемые описанным выше способом, будут иметь сплошную нумерацию на протяжении всего документа. Как мы уже понимаем, это далеко не всегда удобно. Часто хотелось бы сделать так, чтоб, например, в каждом разделе нумерация «теорем» начиналась заново. Для таких целей предусмотрена команда `\newtheorem` с необязательным аргументом. Этот аргумент ставится после двух обязательных и представляет собой имя того счетчика, которому будет подчинен счетчик нашей «теоремы». Пусть, например, в нашем тексте есть не только предложения, но и теоремы (без кавычек), и мы хотим, чтобы нумерация теорем начиналась заново в каждом разделе. Тогда можно написать в преамбуле так:

`\newtheorem{theorem}{Теорема}[section]`

После этого можно будет писать, например, вот что:

Теорема 4.1. *Сумма углов треугольника равна 180° .*

```
\begin{theorem}
Сумма углов треугольника
равна  $180^\circ$ .
\end{theorem}
```

Обратите внимание, что, если «теорема» определена таким образом (со счетчиком, подчиненным другому счетчику), то представление ее номера на печати изменяется: при определении

`\newtheorem{xyz}{abcd}`

(счетчик «теоремы» типа `xyz` подчинен счетчику `abcd`) команда `\the xyz` будет определена как

`\the abcd.\arabic{xyz}`

(если вы хотите, чтобы нумерация «теоремы» представлялась на печати иначе, вы опять-таки можете переопределить `the`-команду).

Наконец, **Л^AT_EX** предоставляет еще одну возможность нумерации определяемых вами «теорем». Предположим, что кроме теорем в вашем тексте есть еще и леммы, и при этом вы хотите, чтобы леммы и теоремы нумеровались совместно: теорема 2.1, теорема 2.2, затем лемма 2.3, затем теорема 2.4, и т. д. Тогда, предполагая, что окружение `theorem` уже определено, как выше, можно определить окружение `lemma` так:

`\newtheorem{lemma}{theorem}[Лемма]`

В этом случае необязательный аргумент команды `\newtheorem` располагается между двумя обязательными; этот аргумент — имя того окружения типа «теорема», совместно с которым будет нумероваться определяемая вами «теорема».

В заключение остается отметить, что команду `\newtheorem` можно использовать или с одним необязательным аргументом, или с другим, но не с обоими вместе.

Глава VIII

Блоки

1. Текст состоит из блоков

Мы уже отмечали, что в процессе верстки TeX не принимает во внимание, как буквы будут выглядеть на печати, а лишь учитывает, сколько места надо отвести на каждый символ. Давайте обсудим этот процесс подробнее.

С точки зрения TeX'a, каждая буква представляет собой блок (английский термин: *box*), т. е. прямоугольник с выделенной *точкой отсчета*; горизонтальная прямая, проходящая через точку отсчета, называется *базисной линией* (английский термин: *baseline*). Блок характеризуется тремя размерами: шириной, высотой и глубиной. См. рисунок, на котором также изображен блок, соответствующий букве *у*.



Когда из букв составляется слова, а из слов — строки, соответствующие отдельным буквам, ставятся рядом так, чтобы их базисные линии были продолжением друг друга. Каждая строка также становится блоком, точка отсчета которого совпадает с точкой отсчета крайнего левого из составляющих ее блоков:

Крокодил \Rightarrow Крокодил

Страницы — это тоже блоки. Когда они составляются из блоков, соответствующих строкам, то эти блоки ставятся таким образом, чтобы точки отсчета были одна над другой, после чего в качестве точки отсчета и базисной линии полученного блока берутся точка отсчета и базисная линия последнего из добавляемых блоков:

~~Когда, наконец, из строк
составляются страницы, то
TeX составляет их из блоков~~

Когда, наконец, из строк
составляются страницы, то
TeX составляет их из блоков

Подведем итоги. При верстке текста *TeX* работает с блоками. Каждая буква также рассматривается как блок, но блоки могут состоять и из больших фрагментов текста. В приведенных выше примерах мы сталкивались с блоками, которые *TeX* делает автоматически; в настоящей главе пойдет речь о командах, предназначенных для создания блоков вручную. Сначала расскажем, какие средства для этого предоставляет нам *Л**T**E**X*, а затем рассмотрим *TeX*'овские команды, имеющие больше возможностей, но более сложные.

2. *Л**T**E**X*'овские команды для генерации блоков

2.1. Блоки из строк

С одной командой для генерации блоков мы уже знакомы: это команда `\mbox`. Эта команда создает блок из текста, набираемого в одну строку. Полученный блок рассматривается *TeX*'ом как одна большая буква:

Проказница мартышка,
осел, козел и косолапый
мишка ...

Проказница мартышка,
`\mbox{осел, козел}` и
косолапый мишка\ldots

В этом примере, кстати, *TeX* никогда не разорвет строку между словами «осел» и «козел», и никогда не сделает переносов в этих словах: при верстке абзаца *TeX* имеет дело не с этими словами по отдельности, а только с блоком, в который входят они оба вместе с пробелом между ними. По той же причине *TeX* не сможет растянуть или сжать пробел между словами «осел» и «козел» для выравнивания строк в абзаце.

Теперь, когда мы знаем, что такое *TeX*'овские блоки, можно признаться, что окружения `picture` и `array` тоже генерируют блоки, и именно поэтому создаваемый ими текст воспринимается *TeX*'ом как одна большая буква.

В аргументе команды `\mbox` может присутствовать все то же, что может быть в обычном тексте в пределах одной строки: математические формулы, команды смены шрифта или присваивания значений

каким-то параметрам, команды для генерации блоков (например, тот же `\fbox`, или даже окружения `picture` или `array`), и т. д. Запрещены в аргументе команды `\fbox` пустые строки или команды `\rag`, выключные математические формулы, окружения, определяющие абзацы специального вида (наподобие `itemize` или `center`), команда `\`` и тому подобные вещи, «не вписывающиеся в строку». Если в аргументе `\fbox` присутствуют команды смены шрифта, изменения каких-то параметров или определения команд, то по выходе из блока все эти изменения забываются, поскольку фигурные скобки, ограничивающие аргумент команды `\fbox`, ограничивают также и группу («глобальные» команды вроде `\setcounter` сохраняют свое действие и по выходе из блока).

Блок, создаваемый командой `\fbox`, имеет ширину, равную «естественной» длине строки текста, являющегося его аргументом. Можно также создать блок из строки текста, ширина которого отлична от ее естественной длины. Для этого используется команда `\makebox`. Эта команда имеет один обязательный аргумент, имеющий такой же смысл, как аргумент команды `\fbox`, и, кроме того, необязательный аргумент — ширину блока, порожденного командой:

Туда и обратно.

Туда `\makebox[5em]{и}` обратно.

Как видите, необязательный аргумент ставится перед обязательным; длина в нем может быть указана, как обычно, либо в какой-либо из ТЕХ'овских единиц, либо через какой-либо параметр со значением длины, возможно — с числовым коэффициентом (см. разд. VII.3). Сам текст, являющийся обязательным аргументом команды `\makebox`, размещается по центру в блоке ширины, указанной в необязательном аргументе. Если указать в необязательном аргументе команды `\makebox` ширину, меньшую естественной длины строки, то текст будет вылезать за края блока; поскольку место, отводимое ТЕХ'ом блоку при верстке, определяется только тем, какова ширина, высота и глубина блока, а не тем, какие размеры реально имеет текст, содержащийся в блоке, при этом может возникать наложение одного текста на другой. Например, размеры и точка отсчета блока, созданного командой `\makebox[1.5em]{123456}`, выглядят, с точки зрения ТЕХ'а, так:

123456

Для ясности мы использовали в этом примере крупный шрифт. А вот как такой «выпирающий за края» блок взаимодействует с окружающим текстом:

текст123456текст

текст`\makebox[1.5em]{123456}`текст

Можно также создать блок заданной ширины, в котором текст будет не центрирован, а прижат к правому или левому краю (полиграфисты говорят: «выключен вправо или влево»). Для этого в команде `\makebox` предусмотрен второй необязательный аргумент — буква `l` для текста, выключенного влево или `r` для текста, выключенного вправо (можно также указать аргумент `c` — тогда текст будет центрирован, так же, как если бы второго необязательного аргумента не было). Пример:

текст	<code>\parindent=0pt</code>
екст	<code>\makebox[10em][r]{текст} \\</code>
кст	<code>\makebox[10em][r]{екст} \\</code>
текст	<code>\makebox[10em][r]{кст} \\</code>
текст	<code>\makebox[10em][c]{текст} \\</code>
	<code>\makebox[10em][l]{текст} \\</code>

Мы установили нулевое значение абзацного отступа, чтобы все строки, включая первую, начинались с самого начала. Кстати, обратите внимание, что у нас получилась верстка с выравниванием без помощи таких вещей, как `tabbing` или `tabular`.

У команды `\makebox` значение ширины блока можно установить равным нулю. Если при этом присутствует необязательный аргумент `l`, то получится блок нулевой ширины, а текст будет выходить за его пределы вправо (и, стало быть, наложится на последующий текст в строке, если таковой присутствует); если присутствует необязательный аргумент `r`, то текст будет выходить влево за пределы блока (и тем самым накладываться на предшествующий текст):

```
текст текст     текст\makebox[0pt][1]{???}текст\\
текст текст     текст\makebox[0pt][r]{???}текст
```

2.2. Блоки из абзацев

Если необходимо создать блок, в котором размещается сверстанный *TeX*'ом абзац текста, то можно воспользоваться командой `\parbox`. У этой команды два обязательных аргумента: первый — длина строк в получаемом абзаце, второй — собственно текст. Например, такой текст

вставили целый абзац	
текста, сверстанного	
В строку по всем <i>TeX</i> 'овским	прерванная строка.
правилам. После этого	
продолжается	

получился следующим образом:

В строку \qqquad

\parbox{4cm}{вставили целый
абзац текста, сверстанного
по всем \TeX'овским правилам.
После этого продолжается}\qqquad
прерванная строка.

Как видите, базисная линия блока, создаваемого командой \parbox, находится в точности посередине текста. Поэтому команду \parbox удобно использовать для включения больших фрагментов текста в математические формулы. Например, формула

$$\int_a^b f'(x) dx = f(b) - f(a)$$

для всех функций f ,
производная которых
интегрируема по Ри-
ману.

получается из такого исходного текста:

```
$$
\int_a^b f'(x) dx = f(b) - f(a)
\parbox{4cm}{для всех функций $f$,
 производная которых интегрируема
 по Риману.}
$$
```

Если дать команду \parbox с необязательным аргументом, то создаваемый ею блок можно расположить относительно строки и по-иному: чтобы вровень с остальной строкой шла самая верхняя строка абзаца (для этого нужен аргумент *t*) или самая нижняя (аргумент *b*) (можно также указать аргумент *c* — тогда блок будет расположен по центру, так же, как если бы необязательного аргумента вообще не было. Необязательный аргумент у этой команды должен идти перед обязательными).

Во втором обязательном аргументе команды \parbox, задающем текст, может присутствовать всё то же, что в обычном тексте, в том числе команды для пробелов по вертикали наподобие \vspace, пустые строки, разделяющие абзацы, выключные формулы и т. п. Абзацы, создаваемые командой \parbox, по умолчанию делаются без абзацного отступа и в режиме \sloppy. Если вы хотите чего-то другого, можно прямо внутри аргумента команды \parbox установить нужное вам значение абзацного отступа, параметра \tolerance и т. п. (см. разд. III.7 по поводу смысла этих параметров).

Наряду с `\ragbox`, существует еще один, довольно экзотический, способ создать блок из абзацев. Именно, существует окружение `minipage` («министрания»), генерирующее блок из текста, расположенного внутри этого окружения; блок состоит из абзацев, ширина которых задается в обязательном аргументе окружения `minipage` (так же, как в команде `\ragbox`); перед обязательным аргументом этого окружения может стоять необязательный: буква `t`, `b` или `c`, причем смысл этого аргумента опять-таки такой же, как в команде `\ragbox`. Основное отличие `minipage` от `\ragbox` в том, что к тексту внутри этого окружения можно делать сноски с помощью команды `\footnote`, причем текст сноски появляется не внизу страницы, а внизу блока, генерируемого окружением `minipage`. При верстке книги, которую вы читаете, это окружение использовалось для печати примеров; придумать разумное применение этой изысканной конструкции вне учебных пособий по *Л*^A_T_E_X'у автору не удалось.

2.3. Текст в рамке; комбинации блоков

В гл. III мы уже упоминали про команду `\fbox`, берущую в рамку фрагмент текста, помещающегося в строку. Наряду с ней есть и команда `\framebox`, относящаяся к ней так же, как `\makebox` относится к `\mbox`: она берет текст в рамку заданного размера, причем текст внутри этой рамки либо центрирует (если необязательного аргумента нет или же задан необязательный аргумент `c`), либо прижимает к правому или левому краю рамки (если задан необязательный аргумент `r` или `l` соответственно). Смысл и расположение обязательных и необязательных аргументов у команды `\framebox` такой же, как и у команды `\makebox`.

Точнее говоря, первый обязательный аргумент команды `\framebox` задает не ширину рамки, а ширину текста, помещаемого в эту рамку. Сама же рамка отделена от текста пробелом, равным значению параметра `\fboxsep`; толщина линий в рамке равна значению параметра `\fboxrule`. Обоим этим параметрам можно обычным образом присваивать новые значения (см. разд. VII.3).

Коль скоро каждый блок, создаваемый *Л*^A_T_E_X'овскими командами, рассматривается *TeX*'ом просто как большая буква, возможны любые, сколь угодно причудливые, комбинации таких «букв». Пусть, например, нам надо взять в рамку абзац текста шириной 6 см, чтобы получилось так:

Внутри TeX'овских блоков может присутствовать не только собственно текст или формулы, но и другие блоки, внутри этих блоков — еще блоки, и так далее. Таким образом, блоки могут быть вложены друг в друга, как матрешки.

Просто поместить этот текст в аргумент команды `\fbox` не получится, поскольку наш текст в одну строку не укладывается, а команда `\fbox`, подобно команде `\mbox`, текстов, не укладывающихся в строку, не переваривает. Поэтому нужно сделать из нашего абзаца блок с помощью команды `\parbox` и этот блок (т. е. уже «букву») передать в качестве аргумента команде `\fbox`:

```
\fbox{%
\parbox{6cm}{%
Внутри TeX'овских блоков может ...
... друг в друга, как матрешки.}}%
```

Обратите внимание на знаки процента, которыми заканчиваются первая и предпоследняя строка. Если бы их не было, то рамка отстояла бы от текста больше, чем надо, так как TeX решил бы, что аргумент команды `\fbox` имеет пробел до и после «буквы», созданной командой `\parbox`. См. с. 17 по поводу использования знака процента для удаления нежелательных пробелов.

2.4. Сдвиги относительно базисной линии

Когда при исполнении команды `\makebox` или `\mbox` TeX создает блок из меньших блоков (каждая буква, как мы помним, — это блок, из букв составляются слова — тоже блоки, и, наконец, блоки могут быть заданы в явном виде, в частности, командой `\mbox`), то блоки эти размещаются в строке таким образом, что все их точки отсчета расположены на одной высоте (иными словами, их базисные линии продолжают одна другую). Можно, однако, сдвинуть блок по вертикали относительно базисной линии. Для этого удобно воспользоваться L^AT_EX'овской командой `\raisebox`. Эта команда требует двух обязательных аргументов. Первый из них — расстояние, на которое сдвигается по вертикали фрагмент текста, второй — сам этот фрагмент текста. Пример:

Слово подскочило в строке. Слово `\raisebox{2pt}{подскочило}` в строке.

Текст, расположенный во втором обязательном аргументе этой команды, должен удовлетворять тем же требованиям, что и аргумент команды `\mbox`: в нем могут быть самые разные *TeX*'овские команды, при условии, что среди них не будет команд типа пустой строки, `\rag`, `\`` и тому подобных, которые «не лезут в строку» (зато в этом тексте, как водится, могут присутствовать любые команды, порождающие блоки, в частности, например, `\parbox`, а уж в ее аргументе оставляйте пустых строк, сколько душе угодно). Если первый обязательный аргумент команды `\raisebox` отрицателен, то текст будет, естественно, не поднят, а опущен. Вот, например, как можно определить команду `\TeX`, печатающую эмблему *TeX*'а:

```
\newcommand{\TeX}{\T\nolinebreak\hspace{- .1667em}\raisebox{-.5ex}{E}\nolinebreak\hspace{- .125em}X}
```

Тут же мы видим и примеры использования отрицательных промежутков для того, чтобы буквы сблизились. Команды `\nolinebreak` нужны, чтобы не случилось разрыва строки посередине эмблемы.

На самом деле команда `\TeX` определяется более экономным способом, который требует меньше машинного времени и памяти, но использует не рассматриваемые нами средства *TeX*'а. Время от времени мы будем приводить определения команд «в переводе с *TeX*'а на *Л**A**T**E**X*», в виде, более понятном читателям этой книги.

Кроме вертикального сдвига блоков, команда `\raisebox` может делать еще одно полезное дело: с ее помощью можно обмануть *TeX*, заставив его считать, что блок, полученный после сдвига, имеет любую заданную нами высоту и глубину, независимо от того, сколько места реально занимает текст. Именно, эта команда может принимать, наряду с обязательными, необязательные аргументы. Между двумя обязательными аргументами можно указать необязательный аргумент — высоту, которую, по мнению *TeX*'а, должен иметь сдвинутый блок. Кроме того, после первого необязательного аргумента может стоять второй — глубина, которую, по мнению *TeX*'а, будет иметь сдвинутый блок. Вот пример:

Строка. Ы	Строка. \\
Вторая	Вторая
Третья строка.	<code>\raisebox{7pt}[1pt][10pt]{Ы}\\</code>

Строка. Ы	Строка. \\
Вторая	Вторая
Третья строка.	<code>\raisebox{7pt}[1pt][10pt]{Ы}\\</code>

Буква І, поднятая на 7 пунктов над строкой, наложилась на первую строку, поскольку в первом необязательном аргументе команды `\raisebox` мы приказали $\text{\TeX}'$ у считать, что блок, образованный поднятой буквой І, имеет высоту всего лишь один пункт (стало быть, возвышается над базисной линией второй строки меньше, чем любая буква), и соответственно \TeX не сделал дополнительного интервала между первой и второй строками. С другой стороны, третья строка отодвинулась от второй, поскольку во втором необязательном аргументе команды `\raisebox` мы велели $\text{\TeX}'$ у считать, что глубина блока, образованного поднятой буквой І, равна целым десяти пунктам, и \TeX послушно оставил дополнительное место, чтобы этот блок не наложился на третью строку!

Иногда разумно использовать команду `\raisebox` даже с нулевым обязательным аргументом, только для того, чтобы менять (в глазах $\text{\TeX}'$ а) высоту и/или глубину блока, не сдвигая его относительно базисной линии. В гл. IX мы увидим пример такого использования этой команды.

2.5. Блоки: дополнительные возможности $\text{\LaTeX}'$ а 2 _{ϵ}

$\text{\LaTeX}'$ 2 _{ϵ} предоставляет ряд дополнительных удобств при работе с командами `\makebox`, `\framebox` и `\ragbox`. Начнем с того, что относится к блокам из строки.

Как мы помним, в первом из необязательных аргументов команды `\makebox` или `\framebox` указывают ширину блока, который должен получиться. В $\text{\LaTeX}'$ е 2.09 эта ширина должна быть задана в явном виде (например, `4em` или `5cm`). В $\text{\LaTeX}'$ е 2 _{ϵ} можно, кроме того, выразить эту ширину через «естественную» ширину текста. Для этого служит команда `\width`. Вот, например, как сделать, чтобы ширина блока, получаемого с помощью `\framebox`, была на 30% больше естественной ширины:

скоросшиватель
скоросшиватель

```
\framebox{скоросшиватель}\[2pt]
\framebox[1.3\width][1]{скоросшиватель}
```

Командой `\width` можно пользоваться только внутри необязательного аргумента `\makebox` или `\framebox`. Не пытайтесь пользоваться ею в качестве параметра со значением длины — ничего хорошего из этого не выйдет.

Второе усовершенствование $\text{\LaTeX}'$ а 2 _{ϵ} по сравнению с $\text{\LaTeX}'$ ом 2.09 относится ко второму необязательному аргументу команды `\makebox`.

и `\framebox`. Кроме известных по *Л*_A*T*_E*X*'у 2.09 букв `g` («прижатый вправо»), `l` («прижатый влево») и `c` («центрированный»), в *Л*_A*T*_E*X*'е 2_ε вторым необязательным аргументом указанных команд может быть и буква `s`, с которой начинаются английские слова `stretched` (растянутый) и `shrunken` (ужатый). Соответственно, при указании такого второго необязательного аргумента текст будет равномерно растянут или сжат до ширины, указанной в первом необязательном аргументе. Если при этом придется превысить предел растяжимости, то появится сообщение `Underfull \hbox`, а если окажется, что превышен предел сжимаемости, то вы увидите сообщение `Overfull \hbox`. Чтобы осмысленно применять `\makebox` и `\framebox` с необязательным аргументом `s`, надо уметь управлять растяжимостью и сжимаемостью промежутков. Как это делать, рассказано в следующем разделе.

Наконец, команда `\parbox` в *Л*_A*T*_E*X*'е 2_ε может принимать не один, а целых три необязательных аргумента. Первый из них — это, как и в *Л*_A*T*_E*X*'е 2.09, буква `t`, `b` или `c`, указывающая расположение базисной линии (сверху, снизу или по центру). Второй необязательный аргумент, идущий непосредственно после первого, указывает высоту, которую должен иметь полученный в результате блок. Наряду с явным указанием размера, можно воспользоваться командой `\height`, обозначающей «естественную» высоту текста, а также командой `\totalheight` (высота плюс глубина). В следующем примере блоки, созданные командой `\parbox`, для наглядности взяты в рамочку (с помощью `\fbox`):

Мы едем,
едем, едем
в далекие
края.

Мы едем,
едем, едем
в далекие
края.

```
\fbox{%
\parbox[b][1.3\height]{2cm}{%
Мы едем, едем, едем
в далекие края.}}\quad
\fbox{%
\parbox[t][1.3\height]{2cm}{%
Мы едем, едем, едем
в далекие края.}}
```

Команду `\height`, так же как и `\depth`, можно использовать только в необязательном аргументе команд `\parbox`, `\framebox` или `\makebox`.

Третий необязательный аргумент — это буква `t`, `b`, `c` или `s`, указывающая, как именно должен быть расположен текст внутри блока. Буква `t` означает «сверху», `b` — «снизу», `c` — «по центру». Если третий необязательный аргумент не указан, то по умолчанию считается, что он совпадает с первым. Если же третьим необязательным аргументом является буква `s`, это означает, что текст будет растянут или сжат в

соответствии с размером, указанным во втором необязательном аргументе. Если вы не позаботитесь о специальных командах, обеспечивающих такую растяжимость или сжимаемость, то получите сообщение об `Overfull'` или `Underfull'`.

3. Команда `\hbox`

Возможности, предоставляемые \LaTeX 'ом для генерации блоков, достаточны для простых приложений, но в более серьезных случаях их не хватает. В этом и следующем разделах мы рассмотрим более гибкие средства, предоставляемые для этой цели непосредственно языком \TeX и макропакетом `Plain \TeX`. Мы не будем пытаться описать все \TeX 'овские команды для генерации блоков (книгу [3] ничто заменить не может), но сообщим тот необходимый минимум сведений, который необходим для модификации \LaTeX 'овских стандартных стилей, о чём пойдет речь в следующей главе. Подчеркнем, что всеми описываемыми в этом и следующем разделах \TeX 'овскими средствами можно пользоваться в \LaTeX 'овских исходных текстах.

Прежде всего давайте вспомним (с. 116), что в каждый момент трансляции исходного текста \TeX находится в одном из трех следующих режимов: горизонтальном (в процессе верстки абзаца), вертикальном (между абзацами), или математическом (в процессе набора математической формулы); при появлении первой же буквы или \LaTeX 'овской команды для генерации блока или линейки (к таковым относятся `\mbox`, `\makebox`, `\fbox`, `\framebox`, окружения `array` или `picture`, а также команда `\rule`²²) \TeX из вертикального режима выходит и начинает очередной абзац.

Одна из основных \TeX 'овских команд для генерации блоков называется `\hbox`. В своем простейшем виде она полностью аналогична \LaTeX 'овской команде `\mbox`, с одним важным отличием: в вертикальном режиме команда `\hbox` не начинает нового абзаца, а только добавляет сгенерированный ею блок (т. е. фактически строку) к уже сверстанной части страницы. Внутри абзаца (в горизонтальном режиме) команда `\hbox` действует точно так же, как и `\mbox`. Вот пример:

²²Но не `\hrule` или `\vrule`: это команды \TeX 'а, а не \LaTeX 'а.

На странице уже присутствует абзац текста. После того, как он кончится, \TeX перейдет в вертикальный режим.

Строка

Еще строка

Только теперь начинается новый абзац.

Сравните с тем, что было бы при использовании \LaTeX 'овской команды \mbox вместо \hbox :

На странице уже присутствует абзац текста. После того, как он кончится, \TeX перейдет в вертикальный режим.

Эти слова сразу начинают новый абзац.

На странице $\text{\hbox\{уже\}}$ присутствует абзац текста. После того, как он кончится, $\text{\TeX\{}}$ перейдет в вертикальный режим.

$\text{\hbox\{Строка\}}$ $\text{\hbox\{Еще строка\}}$
Только

теперь начинается новый абзац.

На странице уже присутствует абзац текста. После того, как он кончится, \TeX перейдет в вертикальный режим.

На странице уже присутствует абзац текста. После того, как он кончится, $\text{\TeX\{}}$ перейдет в вертикальный режим.

$\text{\mbox\{Эти слова\}}$ сразу начинают новый абзац.

3.1. Растижимые интервалы

До сих пор шла речь о важных, но непринципиальных отличиях между \TeX 'овским \hbox и \LaTeX 'овским \mbox . Теперь поговорим о дополнительных возможностях, предоставляемых \TeX 'овской командой.

Команда \hbox «в чистом виде» создает блок, ширина которого равна естественной длине текста, являющегося ее аргументом. Кроме этого, она может создавать блоки любой заданной ширины. Для этого нужно сказать

$\text{\hbox to \langle ширина \rangle \{текст\}}$

Здесь $\langle \text{ширина} \rangle$ должна быть выражена в воспринимаемых \TeX 'ом единицах длины: это может быть, например, 20pt , или 2.3cm , или, например, 0.12textwidth — параметр со значением длины (возможно, с коэффициентом) тоже годится. Между to и обозначением ширины, а также между обозначением ширины и открывающей фигурной скобкой могут быть пробелы — \TeX их проигнорирует²³. Наконец, отсутствие \backslash в слове to не является опечаткой: это не команда, а одно из «ключевых слов» \TeX 'а (подобно ключевым словам plus и minus , с которыми мы вскоре снова встретимся, или width и height , с которыми мы

²³ Пустых строк, однако, быть не должно.

уже встречались в разделе, посвященном линейкам). Давайте опробуем эту новую возможность команды `\hbox`:

Два слова `\hbox to 3cm {Два слова}`

Если вы действительно опробовали этот пример на вашем компьютере, то заметили, что на экране появилось сообщение

`Underfull \hbox`

Дело в том, что пробел между словами «Два» и «слова» не может растянуться настолько, чтобы наш блок имел ширину три сантиметра; в ситуациях, когда пробел насиливо заставляют растянуться больше, чем положено, возникает сообщение об `Underfull`'e, как это было объяснено в разд. III.7.5.

Можно, однако, заставить TeX создать блок требуемой ширины «без скандала». Для этого в том промежутке, который мы хотим растянуть, надо поставить команду `\hfil`:

Два слова	<code>\hbox {Два слова}</code>
Два слова	<code>\hbox {Два \hfil слова}</code>
Два слова	<code>\hbox to 2cm {Два \hfil слова}</code>
Два слова	<code>\hbox to 3cm {Два \hfil слова}</code>
Два слова	<code>\hbox to 4cm {Два \hfil слова}</code>

Если мы не указываем явно ширину блока, а предоставляем TeX'у создать блок «естественной» ширины, то команда `\hfil` никакого действия не оказывает; если промежуток для достижения требуемой ширины надо растянуть, то растяжение на требуемое расстояние будет проведено в том месте, где стоит команда `\hfil`.

Если в аргументе команды `\hbox` присутствует не одна команда `\hfil`, а несколько, то растяжение произойдет на месте каждой из этих команд, причем размер этого растяжения будет распределен между командами `\hfil` равномерно: если необходимо превысить естественную ширину блока на 5 см, а в аргументе команды `\hbox` стоят два `\hfil`, то на месте каждого из них будет добавлен пробел в 2,5 см. Вот пример с несколькими `\hfil`:

Раз два три `\hbox to 4cm{Раз \hfil два \hfil три}`

В частности, если `\hfil` стоит справа или слева от текста, то весь текст будет прижат влево или вправо, поскольку `\hfil` отмечает то единственное место, в котором интервалы могут растягиваться; если же две команды `\hfil` стоят по обе стороны от текста, то текст внутри блока будет центрирован, поскольку дополнительное растяжение поделится между двумя `\hfil` поровну:

Слева	<code>\hbox to 0.7\textwidth {Слева\hfil}</code>
Справа	<code>\hbox to 0.7\textwidth {\hfil Справа}</code>
В центре	<code>\hbox to 0.7\textwidth {\hfil В центре\hfil}</code>

Можно считать, что на месте каждого `\hfil` в строку вставляется пружина; все эти пружины имеют одинаковую жесткость, в свободном состоянии все они имеют нулевую ширину, и все эти пружины могут сколь угодно широко растягиваться.

Наряду с `\hfil` существует команда `\hfill`, также задающая бесконечно растяжимые пробелы, причем эта растяжимость «в бесконечное число раз больше», чем у пробелов, задаваемых `\hfil`. Если в аргументе команды `\hbox` присутствуют `\hfil` и `\hfill` совместно, то все растяжения происходят только за счет «более растяжимых» `\hfill`:

Слово	<code>\hbox to 4cm{\hfil Слово\hfil}</code>
Слово	<code>\hbox to 4cm{\hfill Слово\hfil}</code>
Слово	<code>\hbox to 4cm{\hfil Слово\hfill}</code>

3.2. Лидеры

В оглавлении к этой книге (и ко многим другим тоже) место между названием раздела и номером страницы заполняется рядом из точек. \TeX дает возможность печатать ряды из точек, заполняющие заданный пробел. Для этой цели служит \LaTeX -овская команда `\dotfill`. Она работает так же, как и `\hfill`, с той разницей, что пробел, образующийся в результате действия этой команды, заполняется точками:

А Б `\hbox to 3cm{A\dotfill Б}`

Кроме этого, есть \LaTeX -овская команда `\hrulefill`, которая также действует аналогично команде `\hfill` и при этом заполняет пробел линейкой:

1 _____ 2 _____ 3 `\hbox to 5cm{1\hrulefill
2\hrulefill 3}`

В \TeX нической терминологии такие заполнители называют лидерами (leaders).

На самом деле можно заполнить пробел не только точками или линейкой, но и любым повторяющимся текстом. Вот как это делается. Пусть мы хотим заполнить пробел повторяющимися твердыми знаками. Тогда можно написать так:

1 ЪЪЪЪЪЪЪЪЪъ 2

```
\hbox to 5cm{1\leaders  
\hbox{\b}\hfil 2}
```

Если бы мы хотели, чтоб буквы Ъ шли не вплотную, можно было бы, например, вместо `\hbox{\b}` написать так:

```
\hbox to 2em{\hfil \b\hfil}
```

В общем случае применяйте команду `\leaders` так:

```
\leaders <блок> <\hfil или \hfill>
```

Здесь `<блок>` — это любая ТЕХ'овская команда для генерации блока, например, `\hbox`, с которой мы уже познакомились, или `\vbox` или `\copy`, о которых еще пойдет речь. ЛАТЕХ'овские команды `\mbox`, `\makebox`, `\parbox` и тому подобные применять в этом месте нельзя; если, тем не менее, хочется воспользоваться их возможностями, то их надо «спрятать» в `\hbox`, написав, например,

```
\hbox{\makebox[3em][r]{...}}
```

Между командой для генерации блока и командой `\hfil` или `\hfill` может быть пробел (например, конец строки). Команда `\leaders` работает так: выделяется столько свободного места, сколько получилось бы, если бы стояло просто `\hfil` или `\hfill`, а затем это место заполняется идущими вплотную друг к другу копиями `<блок>` столько раз, сколько этот блок поместится по ширине на выделенное место (если ширина свободного места меньше ширины блока, то ни разу).

С помощью команды `\leaders` можно также задать по своему усмотрению толщину линейки, заполняющей свободное место. Именно, команда `\hrulefill` является по существу сокращением от

```
\leaders\hrule\hfill
```

Если же мы скажем, например,

```
\leaders\hrule height 1pt \hfill
```

то линейка будет иметь толщину 1 пункт, вместо принятых по умолчанию 0.4 пункта. Можно также написать `\hfil` вместо `\hfill`, с очевидными последствиями.

3.3. Клей

Выше мы рассмотрели команды `\hfil` и `\hfill`, которые действуют подобно вставленным в строку пружинам. \TeX позволяет вставлять в строку пружины с самыми разнообразными свойствами. Для этого у \LaTeX 'овской команды `\hspace` надо указать аргумент, содержащий `plus-` или `minus-`компоненту. В разд. III.8.4 мы упоминали об этой возможности, но в тот момент у нас еще не было серьезных примеров. Теперь займемся такими командами вплотную. Если вы скажете

`\hspace{x plus y minus z}`

где x , y и z — длины, то вставите в текст пружину, которая в свободном состоянии имеет длину x , может увеличивать свою длину максимум на y ²⁴ и уменьшать свою длину максимум на z (в отличие от пружин, встречающихся в жизни, может выполняться неравенство $x < z$, и, того пуще, длины y и z могут быть отрицательными, но мы не будем объяснять, как \TeX поведет себя в столь странной ситуации). Здесь `plus` и `minus` — это, как мы помним, очередные ключевые слова $\text{\TeX}'a$, наподобие `to`, `width` и `height`. Если мы создаем блок естественной ширины, то команда `\hspace` с таким аргументом создаст пробел размером x ; если же мы в команде `\hbox` попросим \TeX создать блок, ширина которого отличается от естественной, то для достижения требуемой ширины размеры пробелов будут изменяться. В \TeX нической терминологии эти «пружины» называются **клеем** (Дональд Кнут отмечает, что название «клей» неудачно, но менять его поздно, поскольку оно, по его словам, «уже прилипло»). Длины y и z , указанные после ключевых слов `plus` и `minus`, называются `plus-` и `minus-`компонентами клея. Длина x называется естественным размером клея. С этой точки зрения команда `\hfil` также помещает в строку клей — с бесконечной растяжимостью и нулевым естественным размером.

Опишем более точно, как именно растягивается или сжимается клей при выполнении команды `\hbox to ...`. Для простоты предположим дополнительно, что `plus-` и `minus-`компоненты клея всюду неотрицательны и что в строке отсутствует клей с бесконечной растяжимостью или сжимаемостью (в частности, в строке нет `\hfil'`ов или `\hfill'`ов; про клей с бесконечной сжимаемостью речь пойдет ниже). В этом случае \TeX вычисляет «естественную ширину» блока, складывающуюся из ширин составляющих его элементов и естественных размеров клея, и сравнивает ее с требуемой шириной блока, указанной в команде `\hbox`.

²⁴Если мы заставим ее растянуться больше, чем на y , то получим сообщение `Underfull \hbox`; см. ниже.

после ключевого слова `to`. Если эти две ширины совпадут, то все пробелы будут иметь естественный размер. Если требуемая ширина больше естественной, то `TeX` вычисляет, насколько больше, после чего «разверстывает» эту добавку между всеми пробелами пропорционально величинам `plus`-компонент кляя в этих пробелах.

Вот пример. Предположим, мы создаем блок с помощью команды

```
\hbox to a {A\hspace{Opt plus 2em}%
B\hspace{1cm plus 1em minus 2mm}B}
```

где величина `a` на три миллиметра больше естественной ширины блока АБВ, то пробел между А и Б будет равен 2 мм, а пробел между Б и В — 11 мм, поскольку `plus`-компонента кляя между А и Б в два раза больше, чем `plus`-компонента кляя между Б и В (и никакого другого кляя в строке нет, так что ничего более растянуть нельзя). Если требуемая ширина меньше естественной, то уменьшение длины также разверстывается между всеми элементами кляя пропорционально величинам их `minus`-компонент. Если продолжить аналогию между `TeX`'овским клем и пружинами, то можно сказать, что жесткость пружины при растяжении обратно пропорциональна величине `plus`-компоненты.

В приведенном примере оба пробела в блоке были созданы вручную командой `\hspace`; если же в аргументе команды `\hbox` присутствуют пробелы, то следует учесть, что эти пробелы также, как мы объясняли на с. 113, обладают растяжимостью и сжимаемостью, которая также берется в расчет.

В случае, когда пробелы надо растягивать и треоуемое растяжение блока больше, чем сумма `plus`-компонент всех элементов кляя, на экран и в `log`-файл выдается знакомое вам сообщение `Underfull \hbox`; если пробелы надо уменьшать и величина, на которую надо уменьшить ширину блока, меньше, чем сумма `minus`-компонент всех элементов кляя, то выдается не менее знакомое сообщение `Overfull \hbox`.

Все сказанное относилось к случаю, когда бесконечно растяжимого кляя в аргументе команды `\hbox` нет. Если же таковой присутствует (например, есть команда `\hfil`) и пробелы надо растягивать, то растяжимость кляя с конечными значениями `plus`-компонент утрачивается: соответствующие интервалы будут иметь естественный размер (что бы ни было написано в аргументе команды `\hspace` после `plus`), а все растяжения будут происходить только за счет команд `\hfil`. При этом сообщение об `Underfull` выдаваться не будет, как бы ни растянулись пробелы. Аналогично, если пробелы надо ужимать и присутствует клей с бесконечной сжимаемостью, все уменьшения пробелов произойдут только за его счет, и никогда не будет выдано сообщения об `Overfull`.

Если в аргументе `\hbox` присутствуют команды `\hfil` и `\hfill` совместно, то при необходимости растягивать пробелы учитывается только `\hfill`, в то время как «в бесконечное число раз менее растяжимый» `\hfil` в расчет не берется (не говоря уж о клее с конечной растяжимостью):

Блоки и клей	<code>\hbox to 4cm{\hfil Блоки и клей\hfil}</code>
Блоки и клей	<code>\hbox to 4cm{\hfil Блоки и клей\hfill}</code>
Блоки и клей	<code>\hbox to 4cm{\hfill Блоки и клей\hfil}</code>

На с. 123 мы упоминали о том, что в аргументе команды `\vspace` может, вместо длины `plus-` и/или `minus-`компонентой, стоять команда `\fill` (возможно, с коэффициентом). Как мы теперь понимаем, `\vspace` с таким аргументом также задает бесконечно растяжимый клей. Точнее говоря, `\vspace{\fill}` действует так же, как `\hfill`, в то время как команда `\vspace{0.3\fill}` задает клей, растяжимость которого составляет 30% от растяжимости `\hfill` (тем не менее, этот клей также «бесконечно растяжим» в том отношении, что его присутствие отменяет растяжимость `\hfil`'ов и клея с конечными `plus-`компонентами).

3.4. Бесконечно сжимаемые интервалы

Мы уже два раза упомянули про клей с бесконечной сжимаемостью. Настало время объяснить, какими командами его создавать. Из многих способов укажем один, наиболее часто встречающийся. Команда `\hss` вставляет в строку клей, естественный размер которого равен нулю, и который при этом обладает бесконечной растяжимостью (подобно `\hfil`) и бесконечной сжимаемостью. Типичное применение такого «бесконечно сжимаемого» клея — создавать блоки, ширина которых меньше реального размера текста, или блоки с наложением текстов. В самом деле, посмотрите на такой пример:

Кот Пес	<code>\hbox to 50pt {Кот\hss Пес}</code>
КотПес	<code>\hbox{Кот\hss Пес}</code>
Ko\hss Pec	<code>\hbox to 30pt {Кот\hss Пес}</code>
И\hss Кот	<code>\hbox to 15pt {Кот\hss Пес}</code>
ПесКот	<code>\hbox to Opt {Кот\hss Пес}</code>

Когда мы просим, чтобы ширина блока превышала естественную, команда `\hss` действует так же, как и `\hfil`; когда мы создаем блок с естественной шириной, слова «Кот» и «Пес» стоят вплотную друг к другу (естественная ширина клея, созданного `\hss`, равна нулю). Интересные вещи начинаются, когда мы просим, чтобы ширина была 30 pt

(что меньше естественной). Интервал между словами при этом приходится уменьшить; поскольку его естественный размер равен нулю, то после уменьшения интервал становиться отрицательным, т. е. слово «пес» сдвигается влево (накладываясь на слово «Кот»), причем сдвигается так, чтобы ширина блока (т. е. расстояние от начала слова «Кот» до конца слова «Пес») равнялась требуемым 30 pt. Когда же мы наконец просим, чтобы ширина блока равнялась нулю, слову «Пес» приходится сдвинуться влево настолько, чтобы расстояние от его конца до начала слова «Кот» равнялось нулю — иными словами, кот и пес меняются местами! Заметим, кстати, что точка отсчета всех наших блоков совпадает с точкой отсчета буквы К из слова «Кот».

Еще один пример использования `\hss`: как создать блок, точка отсчета которого будет находиться в правом, а не левом конце текста (мы столкнулись с этой проблемой в гл. V — см. с. 174)? Ответ: надо сказать

```
\hbox to Opt{\hss текст}
```

и все будет в порядке. В самом деле, *текст* имеет ширину, отличную от нуля; чтобы блок имел в итоге нулевую ширину, приходится «уменьшать» тот интервал, где стоит `\hss`; так как интервал уже нулевой, то это уменьшение сводится к тому, что текст сдвигается влево до тех пор, пока расстояние между его концом и точкой отсчета не станет равным нулю — а это и означает, что правый конец текста стал его точкой отсчета. В гл. V мы сказали, что эта проблема решается с помощью ТЕХ'овской команды `\llap`, а теперь мы видим, как ее можно определить:

```
\newcommand{\llap}[1]{\hbox to Opt{\hss #1}}
```

Кстати говоря, именно так она и определяется.

А если сказать

```
\hbox to Opt{текст\hss}
```

то что, спрашивается, будет? Ответ: на сей раз будет уменьшаться интервал после текста; стало быть, сам текст никуда не сдвинется, но после него будет сделан такой «отрицательный пробел», чтобы суммарная ширина равнялась нулю. Иными словами, ТЕХ будет просто считать, что ширина блока равняется нулю — мы обманули ТЕХ, убедив его, что наш текст не занимает места по горизонтали! Для такого обмана (к нему приходится прибегать нередко) предусмотрена специальная ТЕХ'овская команда `\rlap`, определяемая так:

```
\newcommand{\rlap}[1]{\hbox to Opt{#1\hss}}
```

Все это также напоминает ситуацию с командой `\lefteqn`, и напоминает не случайно, поскольку эта команда определяется фактически так:

```
\newcommand{\lefteqn}[1]{\rlap{$\displaystyle #1$}\hss}
```

3.5. Еще раз о линейках

В аргументе команды `\hbox` может присутствовать и ТЕХ'овская команда `\vrule`. Ее ценность в том, что она автоматически создает линейку, высота и глубина которой равна высоте и глубине объемлющего блока (ширина этой линейки будет по умолчанию равна 0,4 пункта). Как объяснялось в разд. III.10, можно задать в явном виде ширину линейки с помощью ключевого слова `width`, высоту — с помощью ключевого слова `height`, а также (о чем в гл. III не говорилось) глубину с помощью ключевого слова `depth` (эти три ключевых слова могут следовать после `\vrule` в произвольном порядке). Приведем один неочевидный пример использования `\vrule` внутри `\hbox`.

Иногда используется следующий способ выделения текста: абзац набирается с некоторым отступом от левого поля, а слева от него, вровень с левым полем, печатается вертикальная линейка.

Предыдущий абзац в исходном тексте выглядел так:

```
\begin{flushleft}
\hbox{%
\vrule\hspace{.5em}\parbox{.9\textwidth}{%
Иногда используется следующий способ выделения текста:
абзац набирается с некоторым отступом от левого поля,
а слева от него, вровень с левым полем, печатается
вертикальная линейка.}}
\end{flushleft}
```

Этот текст, надо думать, нуждается в некоторых пояснениях. Во-первых, в последней строке первая из фигурных скобок закрывает аргумент команды `\parbox`, а вторая — `\hbox`. Во-вторых, мы воспользовались окружением `\flushleft`, чтобы ИТЭХ сам позаботился о разумных отступах до и после абзаца. Параметр `\textwidth` означает, как мы помним, ширину страницы. Теперь рассмотрим, что существует внутри `\hbox`. Сначала там идет линейка, затем отступ на 0.5 em, и затем — огромная «буква», созданная командой `\parbox`. Согласно общему правилу, высота и глубина линейки, заданной командой `\vrule`, равна высоте и глубине объемлющего блока, а они в нашем

случае совпадают с высотой и глубиной «огромной буквы» (ведь кроме нее, другого текста в нашем `\hbox` нет). Тем самым линейка получается как раз нужных размеров, что и требовалось!

Обратите еще внимание на знак процента, завершающий первую строку. Без этого знака получилось бы, что аргумент команды `\hbox` начинается с пробела, соответственно и линейка начиналась бы не с начала, а после пробела (ср. с. 17).

На самом деле в предыдущем примере было бы лучше, если бы правый край выделенного абзаца шел вровень с правым краем остального текста. Чтобы добиться этого, надо первый аргумент команды `\parbox` не взять с потолка, а вычислить. Для этого нам понадобятся переменные со значением длины. Предполагая, что мы определили с помощью `\newlength` переменные `\ширина` и `\разность`, сделаем вот что:

```
\begin{flushleft}
\ширина=\textwidth
\settowidth{\разность}{\vrule\hspace{.5em}}
\addtolength{\ширина}{-\разность}
\noindent\hbox{%
\vrule\hspace{.5em}\parbox{\ширина}{%
{Иногда используется ...
... линейка.}}}
\end{flushleft}
```

Мы воспользовались командой `\settowidth`, чтобы найти размер, который занимает линейка вместе с пробелом. Кстати, если просто написать `\hbox{\vrule\hspace{.5em}}`, то на печати мы ничего не увидим (внутри `\hbox`'а никакого текста нет, так что высота и глубина линейки равна нулю и она тем самым невидима); однако же эта команда создаст пробел, величина которого равна 0.4 пункта плюс 0.5 em. Заключительное замечание: поскольку `flushright`, как и всякое окружение, ограничивает группу, все наши манипуляции с параметрами `\ширина` и `\разность` забудутся по выходе из этого окружения.

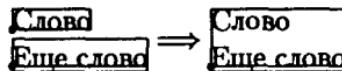
4. Команда `\vbox`

Теперь рассмотрим вторую основную команду TeX'a для генерации блоков — команду `\vbox`. Эта команда создает блок, обрабатывая текст в вертикальном режиме. Вот первый пример:

Слово
Еще слово

```
\vbox{\hbox{Слово}}
      \hbox{Еще слово}}
```

Получаемый блок имеет вид:



Как видите, блоки, создаваемые `\vbox`, ставятся один под другим таким образом, чтобы их точки отсчета лежали на одной вертикальной прямой.

Прежде чем идти дальше, обсудим, что может содержаться в аргументе команды `\vbox`. Там могут присутствовать любые TeX'овские команды, допустимые между абзацами (т. е. в вертикальном режиме): команды `\vspace`, команды смены шрифта, присваивания значений различным параметрам, команды `\newcommand` и `\renewcommand` и т. п. Что же касается команд, которым соответствует что-либо на печати, то будем считать, что из них в аргументе `\vbox` возможны только TeX'овские команды `\hbox`, `\vbox` и `\hrule`, а также `\copy`, о которой речь пойдет позже. В частности, недопустим ни текст, ни L^AT_EX'овские команды `\mbox`, `\parbox`, `\rule` и т. п. Если вам требуется воспользоваться возможностями таких команд, «прячьте» их в `\hbox`, например, так:

```
\hbox{\raisebox{1pt}[2em][3em]{...}}
```

На самом деле в аргументе команды `\vbox` может находиться и обычный текст; при появлении первой же буквы или, скажем, команды `\mbox` или другой команды L^AT_EX'a для генерации блоков TeX переходит в горизонтальный режим, который продолжается до команды, завершающей абзац (`\rag` или пустой строки). Мы не будем вдаваться в детали; для тех приложений, которые мы имеем в виду, достаточно использовать команду `\vbox` так, как было предписано выше.

Когда TeX при выполнении команды `\vbox` составляет блоки друг с другом, он располагает их так, чтобы их базисные линии были, по возможности, на равных расстояниях друг от друга, так что обычно между блоками будет присутствовать дополнительный пробел. С другой стороны, линейки, созданные командой `\hrule`, приставляются к блокам без дополнительного пробела. Чтобы при этом линейка не оказалась вплотную к тексту, удобно в соответствующий блок вставить `\strut`. Следующий пример призван пояснить сказанное:

Неудачно:

Два слова

Лучше так:

Два слова

Неудачно:

```
\vbox{\hbox{Два слова}}
```

```
\hrule
```

Лучше так:

```
\vbox{\hbox{\strut Два слова}}
```

```
\hrule
```

Как обычно, `\vbox` посреди абзаца ведет себя просто как большая буква. Обратите также внимание, что мы не пытались убрать лишний пробел между `\hbox` и `\hrule`: в вертикальном режиме пробелы никакого влияния на текст не оказывают.

Вот еще пример, когда с помощью комбинации блоков и линеек текст берется в рамку:

Текст в рамке

```
\vbox{\hrule
\hbox{\vrule\,,\strut
Текст в рамке\,,\vrule}
\hrule}
```

По-прежнему мы используем `\strut`, чтобы горизонтальные линейки не подходили слишком близко к тексту.

5. Блоковые переменные

В тех случаях, когда один и тот же фрагмент текста (например, фрагмент псевдорисунка, являющийся аргументом команды `\multiput`) используется многократно, бывает полезно сверстать этот фрагмент раз и навсегда, а затем просто повторять его: это сэкономит как количество нажатий на клавиши, так и машинное время. Использование макроопределения в данном случае времени не сэкономит: если мы напишем, например,

```
\newcommand{\abcd}{\parbox{6cm}%
{Когда в товарищах согласья нет,
на лад их дело не пойдет, и выйдет
из него не дело --- только и\’ука.}}
```

то при каждой обработке команды `\abcd` это макроопределение будет заново развертываться, и \TeX будет заново находить переносы и места разрыва строк в одном и том же отрывке из «Лебедя, рака и щуки». Чтобы не заставлять \TeX много раз повторять идентичные операции по верстке текста, надо сделать так. Во-первых, определим «блоковую переменную», которая будет хранить сверстанный фрагмент текста. Это делается с помощью команды `\newsavebox`. Единственный аргумент этой команды — имя новой блоковой переменной, которое должно удовлетворять тем же условиям, что любые имена \TeX 'овских команд: либо `backslash` с одной не-буквой, либо `backslash` с последовательностью букв. Имя новой блоковой переменной не должно совпадать с именем уже существующей команды или переменной длины (если вы попытаетесь нарушить это правило, \LaTeX выдаст сообщение об ошибке). Во-

вторых, присвоим нашей блоковой переменной значение — блок, и будем в дальнейшем этот блок использовать.

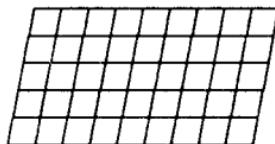
Давайте приведем пример того, как можно этим пользоваться. На с. 179 мы приводили пример псевдорисунка — наклонной решетки, и там же мы отметили, что экономнее было бы сверстать наклонный отрезок раз и навсегда, а затем только повторять его. Теперь мы можем объяснить, как это сделать. Создадим блоковую переменную под названием `\блок`, написав в преамбуле следующее:

```
\newsavebox{\блок}
```

Теперь сверстаем тот текст, который будет храниться в нашей блоковой переменной, и запишем его в эту переменную. Для этих целей используется команда `\sbox` с двумя обязательными аргументами: первый — имя блоковой переменной, второй — текст, который в нее записывается. Итак:

```
\sbox{\блок}{\line(1,5){10}}
```

А теперь можно воспользоваться нашей блоковой переменной. Чтобы напечатать содержимое блоковой переменной, используется команда `\usebox` с одним обязательным аргументом — именем переменной. В нашем случае мы используем блоковую переменную в аргументе команды `\multiput`:



```
\begin{picture}(100,50)
\multiput(0,0)(10,0){10}%
{\usebox{\блок}}
\multiput(0,0)(2,10){6}%
{\line(1,0){90}}
\end{picture}
```

Можно было бы сделать аналогичный трюк и с горизонтальными линейками решетки, но большой экономии это не даст: горизонтальные и вертикальные линейки на псевдорисунках \LaTeX не собирает из отдельных символов, а создает «в один присест» с помощью команды `\rule`, что и так достаточно быстро.

Текст, присутствующий в аргументе команды `\sbox`, будет сверстан в виде блока так, как если бы этот текст был передан в качестве аргумента команде `\hbox` или `\mbox`. Тем самым в аргументе `\sbox` может быть все то же, что может присутствовать в аргументе `\hbox` или `\mbox`. Если команда `\sbox` была дана внутри группы, то по выходе из этой группы содержимое блоковой переменной забудется.

Наряду с командой `\sbox` есть еще и команда `\usebox`, относящаяся к ней примерно так же, как `\makebox` относится к `\mbox`: между первым и вторым обязательным аргументами команды `\usebox` могут присутствовать необязательные аргументы, имеющие тот же смысл и записывающиеся так же, как необязательные аргументы команды `\makebox`. Например, написать

```
\usebox{\пример}[4cm][r]{Слово}
```

все равно, что

```
\sbox{\пример}{\makebox[4cm][r]{Слово}}
```

Наряду с \LaTeX 'овской командой `\usebox` есть похожая на нее, но не идентичная, \TeX 'овская команда `\copy`. Используется она так:

Однажды Лебедь, Рак и Щука ...

```
\sbox{\блок}{Рак}
```

Однажды Лебедь,

```
\copy\блок{} и Щука\ldots
```

Обратите внимание, что при использовании команды `\copy` имя блоковой переменной **не** заключается в фигурные скобки! Различие между `\copy` и `\usebox` такое же, как между `\hbox` и `\mbox`: будучи употребленными внутри абзаца (или, скажем, в аргументе команд `\put`, `\hbox` или `\mbox`), эти две команды действуют совершенно одинаково, а вот будучи употребленным между абзацами, \LaTeX 'овское `\usebox` начинает новый абзац, в то время как \TeX 'овское `\copy` просто подверстывает блок к странице, нового абзаца не начиная. Этую разницу следует иметь в виду, когда вы работаете с командой `\leaders`: выгоднее сверстать блок один раз и записать его в блоковую переменную, а затем в команде `\leaders` писать просто `\copy`. Пример:

```
* * * * * * \usebox{\блок}[1cm]{$*\$}
\hbox to \textwidth
{\leaders\copy\блок\hfil}
```

В этой ситуации по \TeX ническим причинам сказать `\usebox` нельзя.

Глава IX

Модификация стандартных стилей

1. Общие замечания

Эта глава предназначена для тех, кого не удовлетворяет оформление, предлагаемое стандартными стилями $\text{\LaTeX}'$ а. Возможно, вам даже захочется создать свой собственный стиль (пользователь $\text{\LaTeX}'$ а 2 ε сказал бы: класс документов) вместо стандартных `article`, `report` или `book`. Задача эта выполнимая, но для этого надо в деталях знать во-первых, книгу [3], а во-вторых — исходные тексты $\text{\LaTeX}'$ а (они доступны). У читателя настоящей книги таких познаний не предполагается, так что мы предлагаем нечто более скромное, но и более доступное: научиться модифицировать оформление одного конкретного документа.

Прежде чем двигаться дальше — два предупреждения. В этой главе мы расскажем вам, как можно весьма сильно изменить стандартное $\text{\LaTeX}'$ овское оформление: вы научитесь менять по своему усмотрению шрифты в заголовках, интервалы, отделяющие заголовки от текста, и много других подобных вещей. Но вот первое предупреждение: если вы не являетесь профессиональным полиграфистом, применяйте эти знания с осторожностью. Не пытайтесь изменять сразу много разных элементов оформления или резко изменять какие-то параметры: лучше осторожно менять только то, что вам действительно нужно. Учтите: когда дилетант берется за оформление книги, то в девяти случаях из десяти результат бывает достоин сожаления.

Второе предупреждение: стиль оформления записан в специальных «стилевых» или «классовых» файлах, входящих в комплект поставки $\text{\LaTeX}'$ а. НИ В КОЕМ СЛУЧАЕ НЕ МЕНЯЙТЕ НИЧЕГО В ЭТИХ

ФАЙЛАХ: все изменения в стиле надо записывать в преамбуле вашего документа, как это объяснено ниже.

Кое-какие изменения в оформлении документа вы делать уже умеете: например, в гл. IV рассказывалось, как можно, присвоив в преамбуле новые значения некоторым параметрам, изменить размер полей или текста. Однако для более серьезной модификации оформления приходится иметь дело со специальными командами $\text{\LaTeX}'a$, содержащими в своем имени символ \mathbb{C} . Поскольку \mathbb{C} — не-буква, просто так до этих команд не добраться²⁵. Перед тем, как начать с ними работать, необходимо дать команду $\backslash\text{makeatletter}$. Эта команда меняет статус символа \mathbb{C} : \TeX начинает рассматривать его как еще одну букву. После этого можно спокойно работать с командами, содержащими \mathbb{C} в имени; в конце преамбулы надо дать команду $\backslash\text{makeatother}$, восстанавливающую статус \mathbb{C} как не-буквы.

Если вы пользуетесь $\text{\LaTeX}'om 2_{\varepsilon}$, то команды, меняющие оформление, должны идти в преамбуле *после* всех команд $\backslash\text{usepackage}$.

Итак, вот как должен начинаться $\text{\LaTeX}'ovский$ файл, в котором модифицируется стандартное оформление:

В $\text{\LaTeX}'e 2.09$	В $\text{\LaTeX}'e 2_{\varepsilon}$
$\backslash\text{documentstyle...}$	$\backslash\text{documentclass...}$
$\backslash\text{makeatletter}$	$\backslash\text{usepackage...}$
<i>Ваши команды</i>	$\backslash\text{makeatletter}$
$\backslash\text{makeatother}$	<i>Ваши команды</i>
$\backslash\begin{document}$	$\backslash\text{makeatother}$
	$\backslash\begin{document}$

Еще одно замечание для пользователей $\text{\LaTeX}'a 2_{\varepsilon}$. Ниже будут часто встречаться фразы типа «если основной стиль — `book`, то ... ». Применительно к $\text{\LaTeX}'u 2_{\varepsilon}$ вместо «основной стиль» следует говорить «класс документов»; вместо того чтобы повторять эту оговорку десятки раз, давайте условимся, что мы ее сделали сейчас, раз и навсегда (в пределах этой главы).

После всех этих предупреждений пора приступить к делу.

2. Снова о счетчиках

Для начала расскажем об еще двух манипуляциях со счетчиками, которые иногда бывают полезны. До сих пор мы обходили их молчанием,

²⁵Если просто написать $\backslash\text{addtoreset}$ (скоро вы узнаете, что это значит), то \TeX воспримет это как команду \mathbb{C} (имя — из одной не-буквы!), за которой следует текст `addtoreset`.

поскольку команды, используемые для этих манипуляций, содержат `\` в своих именах.

Первый из приемов, о которых пойдет речь, относится к отношению подчинения между счетчиками. Мы знаем, что при создании счетчика с помощью команды `\newcounter` можно задать и счетчик, которому он будет «подчинен» (см. с. 224 и ниже). Но как быть, если уже создан никому не подчиненный счетчик, а мы хотим его кому-то подчинить? Например, за нумерацию сносок отвечает счетчик `footnote`; в стиле `article` этот счетчик определяется как никому не подчиненный, благодаря чему нумерация сносок выходит сплошной в пределах всего документа. Если мы хотим, чтоб нумерация сносок начиналась заново в каждом разделе, то можно в преамбуле (после `\makeatletter`, естественно!) написать так:

```
\@addtoreset{footnote}{section}
```

Первый аргумент команды `\@addtoreset` — имя подчиняемого счетчика, второй — имя подчиняющего.

Разумеется, для того чтобы осмысленно применять описанную команду, надо знать, какие счетчики определены в стандартных стилях и кому они подчинены (или не подчинены). Эта информация содержится в конце раздела.

При ознакомлении с командой `\@addtoreset` может возникнуть искушение написать

```
\@addtoreset{footnote}{page}
```

чтобы сноски нумеровались заново на каждой странице. К сожалению, по ТЕХническим причинам это может не дать желаемого результата: если сноски оказываются на нескольких страницах подряд, то может случиться так, что на второй из этих страниц нумерация сносок начнется не с 1.

Типичный случай использования команды `\@addtoreset` возникает, если основной стиль — `article`. В этом случае в преамбуле часто пишут

```
\@addtoreset{equation}{section}
```

чтобы нумерация уравнений была не сплошной, как предусмотрено стандартом, а начиналась заново в каждом разделе. Разумеется, в этом случае надо будет и переопределить команду `\theequation`.

Вторая обещанная тонкость связана с автоматическими ссылками и `the`-командами. Предположим, что перед исполнением команды `\label{метка}` последним увеличению с помощью `\refstepcounter` подвергался счетчик `abcd`. В гл. VII мы говорили, что при этом команда `\ref{метка}` представит на печати значение этого счетчика «в

соответствии с командой `\theabcd`. Настало время сознаться, что это полуправда. На самом деле команда `\ref` напечатает перед `\theabcd` еще и так называемый «ссылочный префикс» счетчика. Содержимое ссылочного префикса к счетчику `abcd` записано в команде `\p@abcd` (буква `p`, конечно, латинская). В момент создания счетчика эта команда определяется как макроопределение с «пустым» замещающим текстом, так что у таких счетчиков, как `chapter` или `section` в стандартных стилях, ее следов не видно. Можно, однако, переопределить эту команду, чтобы ссылочный префикс реально печатался. Вот пример работы со ссылочным префиксом.

В стандартном стиле `book` вид номера главы и номера раздела определяется следующим образом: команда `\thechapter` определена стандартным образом как `\arabic{chapter}` (такое определение, как мы помним, автоматически производится при создании счетчика), а внешний вид номера раздела определен как

```
\renewcommand{\thesection}{\thechapter.\arabic{section}}
```

Из-за этого номер третьего раздела второй главы печатается в заголовке как 2.3. Предположим, мы хотим, чтобы номера разделов в заголовках не содержали номера главы; тогда можно написать

```
\renewcommand{\thesection}{\arabic{section}}
```

но при этом возникнет другая неприятность. Автоматические ссылки на номер раздела, генерируемые командой `\ref`, теперь дадут на печати одно и то же число 3 как для третьего раздела второй главы, так и для третьего раздела четвертой главы: ведь из команды `\thesection` информация о номере главы ушла! Чтобы справиться с этой неприятностью, поместим утерянную информацию в ссылочный префикс счетчика `section`:

```
\renewcommand{\p@section}{\thechapter.}
```

Теперь будет печататься 2.3 при ссылке на третий раздел второй главы и 4.3 при ссылке на третий раздел четвертой главы: хотя `\thesection` в обоих случаях дает просто 3, печатающийся перед ним `\p@section` обеспечивает печать номера главы и точки. Кстати говоря, именно такой прием применен при подготовке книги, которую вы читаете.

Нам осталось выполнить свое обещание и рассказать, какие счетчики определены в L^AT_EX'овском стандарте и каковы отношения подчинения между ними.

В стиле `article` счетчик `section` определен как

```
\verb"\newcounter{section}"
```

в то время как в стилях `report` и `book`, в которых существуют еще и главы, определяется никому не подчиненный счетчик `chapter` для глав, а счетчик `section` определяется как подчиненный счетчику `chapter`:

```
\newcounter{chapter}
\newcounter{section}[chapter]
```

Остальные счетчики номеров разделов определяются во всех трех стандартных стилях одинаково:

```
\newcounter{part}
\newcounter{subsection}[section]
\newcounter{subsubsection}[subsection]
\newcounter{paragraph}[subsubsection]
\newcounter{ subparagraph}[paragraph]
```

Соответствующие этим счетчикам `the`-команды определены в стиле `article` так:

```
\renewcommand{\thepart}{\Roman{part}}
\renewcommand{\thesection}{\arabic{section}}
\renewcommand{\thesubsection}{\thesection.\arabic{subsection}}
\renewcommand{\thesubsubsection}{%
{\thesubsection.\arabic{subsubsection}}}
\renewcommand{\theparagraph}{%
{\thesubsubsection.\arabic{paragraph}}}
\renewcommand{\thesubparagraph}{%
{\theparagraph.\arabic{subparagraph}}}
```

(мы пишем `\renewcommand`, поскольку все эти `the`-команды уже получили какое-то определение при создании счетчиков).

В стилях `report` и `book`, кроме того, определена `the`-команда для счетчика `chapter` и по-другому определена `\thesection`:

```
\renewcommand{\thechapter}{\arabic{chapter}}
\renewcommand{\thesection}{%
{\arabic{chapter}.\arabic{section}}}
```

За нумерацию сносок отвечает счетчик `footnote`. В стиле `article` этот счетчик определяется как никому не подчиненный:

```
\newcounter{footnote}
```

В стилях же `report` и `book` этот счетчик подчинен счетчику `chapter`, так как в них присутствует еще и команда

```
\@addtoreset{footnote}{chapter}
```

В таком же положении, как счетчик `footnote`, находится и отвечающий за нумерацию формул счетчик `equation`: в стиле `article` он определен как никому не подчиненный, а в стилях `report` и `book` он подчинен счетчику `chapter`. Однако же в стилях `report` и `book` переопределется `\theequation`:

```
\renewcommand{\theequation}%
{\thechapter.\arabic{equation}}
```

Наконец, счетчики `figure` и `table`, отвечающие за нумерацию плавающих иллюстраций и таблиц соответственно, устроены точно так же, как счетчик `equation`: в стиле `article` они никому не подчинены, а в двух других стандартных стилях они подчинены счетчику `chapter` и соответствующие `the`-команды определены как

```
\renewcommand{\thefigure}%
{\thechapter.\arabic{figure}}
```

(аналогично для `table`).

Остались еще счетчики, связанные с нумерованными перечнями. Как объяснялось в разд. VII.2.5, эти счетчики, в зависимости от уровня вложенности `enumerate`, называются `enumi`, `enumii`, `enumiii` и `enumiv`. Все эти счетчики, естественно, последовательно подчинены друг другу, а их ссылочные префиксы определены так (это — единственный случай, когда в стандарте используются нетривиальные ссылочные префиксы):

```
\renewcommand{\p@enumii}{\theenumi}
\renewcommand{\p@enumiii}{\theenumi(\theenumii)}
\renewcommand{\p@enumiv}{\p@enumiii\theenumiii}
```

3. Разделы документа

Здесь мы расскажем о том, как менять оформление разделов документа, предписываемое `LATEX`'овским стандартом.

3.1. Что нумеровать и что включать в оглавление

`LATEX` определяет, какие разделы документа нумеровать, а какие — нет, исходя из двух вещей: «уровня вложенности» раздела и значения счетчика `secnumdepth`. Раздел документа получает номер, если уровень его вложенности меньше или равен значению `secnumdepth` (разумеется, все

сказанное относится к случаю, когда \LaTeX 'овская команда для раздела документа дана без звездочки — иначе нумерации не будет заведено). Уровни вложенности в стандартных стилях приведены в табл. IX.1. Стало быть, если мы хотим, чтобы самый мелкий из нумеруемых разделов

Таблица IX.1. Уровни вложенности разделов документа

Название раздела	Уровень
<code>\section</code>	1
<code>\subsection</code>	2
<code>\subsubsection</code>	3
<code>\paragraph</code>	4
<code>\ subparagraph</code>	4

был `\subsection`, то надо написать в преамбуле документа команду
`\setcounter{secnumdepth}{2}`

Что касается команд `\chapter` и `\part`, то соответствующие разделы документа будут нумероваться тогда и только тогда, когда значение `secnumdepth` является неотрицательным числом.

Какие разделы включать в оглавление, определяется другим счетчиком, а именно `tocdepth`: информация о разделе документа вносится в оглавление в том и только том случае, если уровень вложенности раздела меньше или равен значения этого счетчика (и раздел вводится командой без звездочки).

3.2. Модификация команд, задающих разделы

Теперь рассмотрим, что надо делать, чтобы более серьезным образом изменить оформление разделов. Для этого надо переопределить команды `\section`, `\subsection` и т. п., а чтобы научиться их должным образом переопределять, надо узнать, как эти команды определены в стандартных \LaTeX 'овских стилях.

Почти все команды для создания разделов документа определяются в стилевых (в $\text{\LaTeX}'e 2e$ — классовых) файлах через команду `\@startsection`. Например, команда `\section` определяется (не буквально, а по существу) так:

```
\newcommand{\section}{\@startsection{section}{1}{0pt}%
{-3.5ex plus -1ex minus -.2ex}{2.3ex plus .2ex}%
{\Large\bf}}
```

Как видите, в этом определении у команды `\f@section` указаны шесть аргументов, в которых закодированы различные параметры оформления раздела. Разберем последовательно, что эти аргументы означают.

Первый из аргументов (в нашем случае `section`) — это «внутреннее имя», под которым L^AT_EX будет узнавать определяемый тип разделов документа. Если вы решили использовать команду `\f@section` с «нестандартным» первым аргументом (скажем, `abcd`), то заодно придется определить счетчик с именем `abcd`, который будет отвечать за нумерацию разделов, а также команду `\l abcd`, которая будет отвечать за сбор материала для оглавления (см. подробности в разд. 4) и команду `\abcdmark`, отвечающую за передачу информации для колонтитулов (см. разд. 6). Без надобности командой `\f@section` с «нестандартным» первым аргументом лучше не пользоваться.

Второй аргумент (в нашем случае 1) — это тот самый «уровень вложенности» раздела, о котором шла речь выше.

Третий аргумент задает отступ заголовка от левого поля (в нашем случае этот отступ равен нулю).

Четвертый аргумент команды `\f@section` (в нашем случае это `-3.5ex plus -1ex minus -.2ex`) задает величину вертикального отступа, оставляемого перед заголовком. Точнее говоря, вертикальному отступу равен не сам четвертый аргумент, а его абсолютная величина (при определении отступа знаки — отбрасываются), а знак — означает, что первый абзац нашего раздела будет печататься без абзацного отступа, как и оформляются разделы в стандартных L^AT_EX'овских стилях. Если задать эти расстояния как положительные числа, то абзацный отступ в первом абзаце подавляться не будет. Тот факт, что отступ перед началом раздела имеет `plus-` и `minus-`компоненту, означает, как водится, что этот пробел обладает растяжимостью и сжимаемостью (см. с. 122): после `plus` указано, насколько, самое большое, можно растянуть дополнительный интервал перед заголовком раздела, а после `minus` — насколько, самое большое, можно его ужать.

Пятый аргумент (в нашем случае `2.3ex plus .2ex`) задает величину вертикального отступа *после* заголовка раздела, а тот факт, что он положителен, означает, что заголовок раздела печатается на отдельной строке (или строках, если в строку он не умещается). Если он будет отрицательным, то заголовок раздела будет печататься не на отдельной строке, а в подбор; значение пятого аргумента `\f@section` будет означать при этом (после отбрасывания знака минус, естественно) величину дополнительного горизонтального отступа между заголовком раздела и продолжающим его текстом из первого абзаца раздела.

Пятый аргумент `\@startsection` тоже, как видите, может содержать `plus-` и/или `minus`-компоненту.

Наконец, шестой аргумент команды `\@startsection` задает стиль оформления заголовка. Точнее говоря, в этом аргументе записан текст и/или команды, которые будут вставлены перед заголовком раздела. В нашем случае этот аргумент содержит только команды `\Large\bf`, задающие шрифт, которым заголовок будет напечатан (на последующий текст эта смена шрифта не повлияет, поскольку команды, указанные в шестом аргументе `\@startsection`, будут выполняться внутри группы).

В $\text{\LaTeX}^e 2_e$ шестой аргумент записывается с помощью более «прогрессивных» команд: `\normalfont \Large \bfseries`.

Приведем пример того, как можно изменить стандартное оформление. Пусть нам хочется, чтобы команда `\section` порождала раздел документа, оформленный таким образом:

- Номера разделов печатаются римскими цифрами.
- Перед номером раздела стоит знак §.
- Номер раздела и его заглавие печатаются прямым светлым шрифтом размера `\Large`.
- Абзацный отступ в первом абзаце не подавляется.

Как этого добиться? Для начала переопределим команду `\thesection`, определяющую, в каком виде будет представлен на печати номер раздела:

```
\renewcommand{\thesection}{\Roman{section}}
```

А теперь переопределим и саму команду `\section` (в преамбуле, после `\makeatletter`) следующим образом:

```
\renewcommand{\section}{\@startsection{section}{1}%
{\parindent}{3.5ex plus 1ex minus .2ex}%
{2.3ex plus .2ex}{\Large\bf S}}
```

Мы воспользовались `\renewcommand`, поскольку команда `\section` ранее уже была определена. В четвертом аргументе мы убрали знаки `-`, чтобы не подавлять абзацный отступ, а в третьем — задали отступ заголовка от левого поля, равный абзацному отступу (причина этого не программистская, а эстетическая: некрасиво, когда первый абзац раздела идет с отступом, а заголовок начинается вплотную к левому полю). Прочие

параметры, задающие размеры отступов, мы оставили такими же, как в стандартной \LaTeX 'овской команде: они достаточно разумны, и незачем их трогать, если на то нет особых причин. В $\text{\LaTeX}_2\epsilon$ последний аргумент лучше, конечно, задать в виде $\text{\normalfont Large S}$.

Как видите, для того, чтобы модифицировать оформление разделов, надо знать размеры стандартных \LaTeX 'овских параметров оформления, наподобие отступа перед или после заголовком. Для команды \section мы привели их выше, а для остальных стандартных команд

Таблица IX.2. Стандартное оформление разделов

	Отступ	Перед	После
\subsection	Opt	-3.25ex plus $-1\text{ex minus}.2\text{ex}$	1.5ex $\text{plus}.2\text{ex}$
\subsubsection	Opt	-3.25ex plus $-1\text{ex minus}.2\text{ex}$	1.5ex $\text{plus}.2\text{ex}$
\paragraph	Opt	$3.25\text{ex plus}1\text{ex}$ $\text{minus}.2\text{ex}$	-1em
\ subparagraph	\parindent	$3.25\text{ex plus}1\text{ex}$ $\text{minus}.2\text{ex}$	-1em

рубрикации они собраны в табл. IX.2. В ней «отступ» означает отступ заголовка от левого поля, «перед» — отступ перед заголовком (если этот отступ отрицательный, то абзацный отступ в первом абзаце будет подавлен), «после» — отступ после заголовка (если это отрицательное число, то заголовок печатается в подбор). Уровни вложенности разделов см. в табл. IX.1, а что до «внутреннего имени» раздела (первый аргумент команды \@startsection), то оно у всех стандартных команд рубрикации совпадает с их именем (section для команды \section и т. п.). Шестой аргумент \@startsection содержит в стандартных определениях только команды переключения шрифта; при необходимости заинтересованный читатель воспроизведет его самостоятельно.

Шестой аргумент команды \@startsection позволяет автоматически добавлять текст перед номером раздела. К сожалению, не существует доступного читателям этой книги способа заставить \LaTeX автоматически добавлять текст после номера, хотя такая потребность порой возникает (например, хочется, чтобы в заголовках разделов после номеров стояли точки, чего стандартное \LaTeX 'овское оформление не предусматривает). Можно создать «русифицирующий стиль» (для $\text{\LaTeX}_2\epsilon$ — стилевой пакет), в котором этот недостаток устранен.

Можно заставить \LaTeX ставить точки после номеров, написав

```
\renewcommand{\thesection}{\arabic{section}.}
```

но после этого автоматические ссылки на раздел, сгенерированные с помощью команды `\ref`, также будут заканчиваться точками, что нелепо.

Оформление глав отличается от оформления остальных разделов тем, что слово «Глава» и номер главы печатаются на отдельной строке. С помощью команды `\startsection` определить такой раздел нельзя, поэтому главы определяются в \LaTeX -овских стилях иначе. Не будем вдаваться в подробности, как именно, а вместо этого рассмотрим единственно важный для нас вопрос: как можно менять оформление глав.

Большая часть оформления главы задается в определении команды `\makechapterhead`, так что для модификации оформления именно ее и надо переопределять. Рассмотрим, как `\makechapterhead` определяется в стандарте. Этой команде передается один аргумент — заголовок главы. В переводе с \TeX 'а на \LaTeX определение выглядит так (не забудьте, что #1 — это аргумент, т. е. текст заголовка).

```
\newcommand{\makechapterhead}[1]{% Начало макроопределения
  \vspace*{50 pt} % Пустое место вверху страницы
  {\parindent=0pt
    \raggedright \huge\bf
    \chapapp{} % \chapapp печатает слово "Глава" (см. ниже)
    \thechapter \par % номер главы - в отдельной строке
    \vspace{20pt} % между словом "Глава" и ее заголовком
    \Huge \bf #1\par % заголовок главы
    \nopagebreak % чтобы не оторвать заголовок от текста
    \vspace{40 pt} % между заголовком и текстом
  }% конец группы.
}% конец макроопределения
```

Разберем эту «программу». Первая команда `\vspace*` оставляет пустое место вверху страницы (поскольку главы начинаются с новой страницы). Далее печатается слово «Глава», ее номер и заголовок главы; поскольку заголовок может не поместиться в строку, надо предусмотреть, какие будут при этом параметры верстки абзаца. Как видите, устанавливается нулевое значение абзацного отступа и отсутствие выравнивания по правому краю; чтобы такой режим не распространялся на дальнейший текст, соответствующие команды даны внутри группы. Внутри этой же группы определен шрифт, которым будет печататься заголовок.

Команда `\chapapp` печатает слово «Chapter», «Глава»... — одним словом, то, как в вашем документе называются главы. Точнее говоря,

по умолчанию эта команда работает так же, как `\chaptername`, а после команды `\appendix` (если таковая есть в вашем файле) начинает работать как `\appendixname` (см. с. 156). Если в вашем тексте все главы называются одинаково, то при переопределении `\@makechapterhead` можно не мудрить, а прямо заменить `\@chapapp` на `Глава`. Не забудьте только оставить пробел между этим словом и номером главы (выше это было сделано с помощью обычного трюка с парой фигурных скобок).

Остальное в приведенном выше определении разъяснений, надо думать, не требует. Скорее всего, вы захотите в этой команде изменить шрифт, которым печатается заголовок главы, или же интервалы, отделяющие заголовок от остального текста. Чтобы изменить вид, в котором представляется на печати номер главы, надо, как водится, переопределить команду `\thechapter`. Можно при желании задать и какое-нибудь более сложное оформление главы с помощью блоков и линеек — все зависит от вашей фантазии и вкуса!

Надо еще сказать, что мы немного обманули читателя: на самом деле в стандартных стилях команда `\@makechapterhead` определена таким образом, что, если значение счетчика `secnumdepth` отрицательно, то команды, записанные в строках с пятой по седьмую этого определения, не исполняются (т. е. номер главы не печатается). В нашем «переведенном на L^AT_EX» определении эти команды будут исполняться всегда, вне зависимости от значения `secnumdepth`; если вы переопределяете `\@makechapterhead` и не хотите, чтобы главы нумеровались, просто удалите соответствующие строки из определения.

- ε В L^AT_EX'е 2_ε, естественно, в этом определении применяются другие команды смены шрифта. Кроме того, если в классе `book` вы переопределите `\@makechapterhead`, отталкиваясь от этого определения, то главы, заданные как `\chapter` без звездочки, будут нумероваться, невзирая на команды `\frontmatter` и `\backmatter` (но ε вы ведь сможете поставить две-три звездочки самостоятельно?).

За оформление заголовка главы, определенной командой `\chapter` со звездочкой, отвечает команда `\@makeschapterhead`. Ее определение в стандарте аналогично определению `\@makechapterhead`, с тем отличием, что из него удален фрагмент, отвечающий за печать номера:

```
\newcommand{\@makeschapterhead}[1]{%
  \vspace*{50 pt}%
  {\parindent=0pt \raggedright
   \Huge \bf #1\par
   \nopagebreak
   \vspace{40 pt}}}
```

Кроме оформления заголовка с оформлением глав можно делать еще две вещи. Во-первых, главы начинаются либо просто с новой страницы (как в стиле `report`), либо с новой нечетной страницы (как в стиле `book`); у вас может возникнуть желание повлиять на этот выбор (если вы пользуетесь $\text{\LaTeX}'ом 2\epsilon$, то к вашим услугам опции `openright` и `openany`, но в $\text{\LaTeX}'е 2.09$ их нет). Во-вторых, по умолчанию абзацный отступ в первом абзаце главы подавляется; вы можете захотеть сделать так, чтобы он не подавлялся. Чтобы решить обе эти проблемы, надо переопределить уже саму команду `\chapter`, а для этого надо знать, как она определяется в стандарте. Вот соответствующее определение, опять в переводе на \LaTeX с $\text{\TeX}'а$, в стиле `book`:

```
\newcommand{\chapter}{\cleardoublepage
  \thispagestyle{plain}%
  \global\c@topnum=0
  \c@afterindentfalse
  \secdef\chapter\@chapter}
```

Разбирать это определение мы не будем, чтобы не запутаться в некоторых слишком хитрых $\text{\TeX}'овских$ и $\text{\LaTeX}'овских$ конструкциях, а просто скажем две вещи:

- Чтобы глава начиналась не обязательно с нечетной страницы, замените в этом определении `\cleardoublepage` на `\clearpage`.
- Чтобы не подавлялся абзацный отступ в первом абзаце главы, замените `\c@afterindentfalse` на `\c@afterindenttrue`.

Что касается команды `\part` («часть»), то она отличается тем, что заголовок части занимает отдельную страницу. Если вы решили изменить стандартное оформление таких «частей», то проще всего оформить соответствующие две-три страницы-заголовка вручную.

4. Оглавление, список иллюстраций и прочее

Автоматическая сборка оглавления — многоэтапный процесс. Сначала материал для оглавления (заглавия разделов и номера соответствующих страниц) записывается в специальный файл с тем же именем, что и у основного файла, и расширением `toc` (в нормальных условиях эта запись обеспечивается командами `\chapter`, `\section` и т. д.); при следующем запуске $\text{\LaTeX}'а$ этот `toc`-файл считывается (с помощью команды `\input`), команды, записанные в него, исполняются, и в результате происходит фактическая верстка оглавления. Аналогичным образом составляется список иллюстраций и список таблиц (при этом информация

записывается в файлы с расширениями `lof` или `lot` соответственно). Давайте научимся влиять на этот процесс.

Сначала расскажем, как составлять оглавление полностью вручную, игнорируя его автоматическую сборку, обеспечиваемую командами типа `\section`. Итак, предположим, что все команды `\section`, `\chapter` и т. п. даны в исходном тексте в варианте со звездочкой, и посмотрим, как можно самому создать оглавление.

Команда `\addtocontents` служит для записи в `toc-` (соответственно, `lof-` или `lot-`) файл любого текста и любых \TeX 'овских команд. У этой команды два обязательных аргумента. Первый из них должен быть `toc`, `lof` или `lot`, в соответствии с тем, в какой из файлов с оглавлениями вы пишете свой текст. Второй аргумент — то, что вы хотите записать в файл. Если, например, вам взбрело в голову внести в оглавление к вашей книге текст «У попа была собака» (не будем выяснять, зачем), то можете написать:

```
\addtocontents{toc}{У попа была собака.}
```

Если после этого запустить \LaTeX два раза, то вы увидите в оглавлении свой текст (после первого раза он только попадет в `toc`-файл, а при втором запуске `toc`-файл с этим текстом будет обработан).

С помощью команды `\addtocontents` можно записывать в оглавление не только всякие глупости. Если, например, вы хотите в каком-то месте оглавления провести горизонтальную линейку шириной во всю страницу, то можно написать

```
\addtocontents{toc}{\hrule}
```

и в оглавлении появится линейка. Имейте только в виду, что в аргументе `\addtocontents` необходимо защищать хрупкие команды с помощью команды `\protect` (см. с. 164). В случае с `\hrule` мы обошлись без `\protect`, так как эта команда не хрупка, но вообще, если есть сомнения, то лучше команду защитить. Напомним, что `\protect` действует только на непосредственно следующую команду и что команды смены шрифта в защите с помощью `\protect` не нуждаются. Приведем пример более разумного применения `\addtocontents`, в котором требуется `\protect`. Пусть вы не хотите, чтобы какая-то из строк в оглавлении начинала новую страницу. Тогда надо перед командой, порождающей эту строку оглавления (обычно таковой будет команда наподобие `\section`) написать в своем файле вот что:

```
\addtocontents{toc}{\protect\nopagebreak}
```

В результате в `toc`-файл запишется команда `\poragebreak`, и нежелательный разрыв страницы в оглавлении будет предотвращен. Если опустить `\protect`, то получится весьма непонятное сообщение об ошибке.

Чтобы составить полноценное оглавление, надо иметь возможность записать в `toc`- (соответственно, `lof`- или `lot`-) файлы не только текст, но и номер той страницы, к которой этот текст относится. Это делается с помощью команды `\addcontentsline`, имеющей такой синтаксис:

`\addcontentsline{тип_файла}{тип_записи}{текст}`

Здесь *тип_файла* — это `toc`, `lof` или `lot`, *текст* — тот текст, который будет записан в оглавление (например, команда `\section` в стандартном стиле `article` в качестве этого текста передает название раздела и его номер; подробности см. ниже). Наконец, *тип_записи* определяет, каким образом будет обрабатываться этот текст при чтении файла с оглавлением. Именно, если второй аргумент в команде `\addcontentsline` был `abcd`, то, когда при следующем запуске `LATEX`'а будет читаться `toc`- (соответственно, `lof`- или `lot`-) файл, будет исполнена команда `\l@abcd` с двумя аргументами, первый из которых — текст, записанный в третьем аргументе команды `\addcontentsline`, а второй — номер страницы, на которую попала ваша команда `\addcontentsline`. Например, если в файле было написано

`\addcontentsline{toc}{abcd}{0 слонах} (*)`

и если эта команда попала на страницу 95, то при следующем запуске `LATEX`'а в процессе чтения `toc`-файла будет исполняться команда

`\l@abcd{0 слонах}{95}`

Разумеется, чтобы при этом не получилось сообщения об ошибке, надо, чтобы команда `\l@abcd` была определена. Стало быть, в преамбуле должно присутствовать ее определение. Если мы хотим, чтобы запись (*) в исходном файле порождала в оглавлении запись вида

О слонах.....95

то в преамбуле надо написать вот что:

`\newcommand{\l@abcd}[2]{\hbox to\textwidth{\#1\dotfill \#2}}`

Чтобы при этом страница в оглавлении была указана верно, необходимо команду `\addcontentsline` разместить непосредственно после команды `\section*` (иначе есть опасность, что они попадут на разные страницы).

Если в третьем аргументе команды `\addcontentsline` присутствуют «хрупкие» команды, то их следует, как водится, защитить командой `\protect`; если, с другой стороны, в нем записана `\the`-команда, соответствующая какому-то счетчику, то в `toc`-файл будет записано печатное представление значения этого счетчика по состоянию на тот момент, когда выполнялась `\addcontentsline`. Таким способом можно, например, записать в оглавление номер текущего раздела документа: достаточно сказать

```
\addcontentsline{toc}{abcd}%
{\thesection. 0 слонах}
```

Теперь рассмотрим, как именно собирают оглавление стандартные команды наподобие `\chapter` или `\section`. Делают они это также с помощью `\addcontentsline`, при этом ее второй аргумент («тип записи») будет `section` для команды `\section`, `subsection` для команды `\subsection`, — одним словом, «внутреннее имя», под которым `LATEX` знает тип разделов документа (напомним, что внутреннее имя передается в качестве первого аргумента команде `\@startsection`). Стало быть, для модификации стиля оформления строк оглавления, соответствующих `\section`, надо переопределять команду `\l@section`, для модификации строк оглавления, соответствующих `\subsection`, надо переопределять `\l@subsection`, для модификации строк оглавления, соответствующих `\chapter`, надо переопределять команду `\l@chapter`, и т. д. Чтобы было понятно, как их переопределять, рассмотрим, как они определены в стандарте.

В стиле `book` команда `\l@section` определена так:

```
\newcommand{\l@section}{\@dottedtocline{1}{1.5em}{2.3em}}
```

Смысл трех аргументов команды `\@dottedtocline` таков. Первый аргумент — «уровень вложенности» элемента оглавления. Если этот уровень превышает значение счетчика `tocdepth`, то команда `\@dottedtocline` ничего в оглавлении не печатает. Второй аргумент — отступ строки оглавления от левого поля. Третий аргумент определяет, сколько места в строке оглавления `TeX` отведет на номер раздела. Результат выглядит на печати так: после отступа, указанного во втором аргументе, печатается номер раздела, затем, отступя от *начала* этого номера столько, сколько сказано в третьем аргументе, печатается заглавие раздела. Это сделано для того, чтобы заглавия всех разделов печатались в оглавлении одно под другим. После заглавия идут «лидеры» — ряд точек до завершающей строки номера страницы. Если заглавие в строку не укладывается, то оно обычным образом будет перенесено на следующую строку

(если есть какие-то неясности, загляните в оглавление к этой книге). Из сказанного следует, что, слишком длинный номер раздела может в оглавлении наложитьться на заглавие. Средство борьбы с этим — переопределить команду `\l@section` (или `\l@subsection ...`), увеличив должным образом второй аргумент команды `\@dottedtocline`.

Параметры оформления элемента оглавления, задаваемого командами, определенными через `\@dottedtocline`, можно менять. Именно, размер места, отводимого на номер страницы, задается значением команды `\@pnumwidth`, которую можно переопределить. В стиле `book` эта команда определена как

```
\newcommand{\@pnumwidth}{1.55em}
```

и соответственно на номер страницы отводится 1.55em места. Если мы хотим, чтобы на номер страницы отводилось 2em, надо написать

```
\renewcommand{\@pnumwidth}{2em}
```

Еще одна команда, значение которой отвечает за оформление оглавления, — это `\@tocrmarg`. Если запись в оглавлении занимает более одной строки, то значение этой команды задает отступ от правого поля, который будет у всех строк, кроме той последней, что завершается номером страницы. Если вы хотите, чтобы размер этого отступа равнялся 3em, напишите так:

```
\renewcommand{\@tocrmarg}{3em}
```

Хотя `\@pnumwidth` и `\@tocrmarg` используются для задания размеров, они не являются параметрами со значением длины; запись наподобие `\@tocrmarg=4em` приведет к ошибке!

Наконец, регулировать густоту точек-«лидеров» можно, если переопределить команду `\@dotsep`. В стиле `book` она определена как

```
\newcommand{\@dotsep}{4.5}
```

Если вы хотите, чтобы точки шли погуще, попробуйте переопределить ее, заменив 4.5 на число поменьше (число может быть дробным, в нем можно использовать как десятичную запятую, так и десятичную точку):

```
\newcommand{\@dotsep}{3.9}
```

Напрашивающаяся запись `\@dotsep=3.9` приведет к ошибке.

Команды `\l@subsection` и «более мелкие» определяются в стандарте так же, как `\l@section`, отличаются только аргументы команды `\@dottedtocline`. Для ориентировки мы собрали значения этих параметров в табл. IX.3.

Таблица IX.3. Стандартные определения `\@-команд` (стиль `book`)

	Аргументы <code>\@dottedtocline</code>		
<code>\@subsection</code>	2	3.8em	3.2em
<code>\@subsubsection</code>	3	7.0em	4.1em
<code>\@paragraph</code>	4	10em	5em
<code>\@ subparagraph</code>	5	12em	6em

Теперь рассмотрим, как в стандарте определяются записи в оглавлении, соответствующие самым крупным разделам. Вот (в адаптированном виде, как водится) определение команды `\@chapter` из стандартного стиля `book`. Чтобы было понятно, о чем идет речь, напомним, что в этом определении на место #1 подставляется информация о номере (если главы нумеруются) и заглавии главы, а на место #2 — номер страницы.

```
\newcommand{\@chapter}[2]{%
  \pagebreak[3]\vspace{1em plus 1pt}%
  \tempdima=1.5em % место для номера главы
  % Дальнейшее происходит внутри группы...
  \rightskip=\tempdima % см. ниже
  \leftskip=\tempdima % см. ниже
  \bf % установить полужирный шрифт
  \noindent % начать абзац без отступа...
  \hspace{-\leftskip}% и с левого поля.
  #1\nobreak\hfil\nobreak
  \rlap{\makebox[\tempdima][r]{#2}}\par
  \pagebreak[3]%
} % конец этой группы
} % конец определения
```

Определение, как видите, длинное и сложное, к тому же автору не удалось полностью изгнать из него не упоминавшиеся ранее \TeX 'овские конструкции. Приведено оно здесь не для того, чтобы вы самостоятельно создавали подобные определения «с нуля» (для этого надо знать про \TeX больше, чем написано в этой книге), но чтобы вы при необходимости смогли в нем кое-что осторожно изменить. Чтобы было понятно, что и как менять, разберем определение по порядку. В начале встречаются не рассматривавшиеся нами параметры `\leftskip` и `\rightskip`. Эти параметры со значением длины (по умолчанию они равны нулю) имеют следующий смысл: все строки абзаца начинаются с отступом `\leftskip` от левого поля и кончаются с отступом `\rightskip`.

от правого поля. У нас `\rightskip` устанавливается равным длине, записанной в определении команды `\@pnumwidth` (именно столько места будет отведено на номер страницы), а `\leftskip` устанавливается равным значению переменной со значением длины `\@tempdima`, определяющей, сколько места будет отведено на номер главы. Между `\@pnumwidth` и `\@tempdima` есть существенная разница. Команда `\@pnumwidth` всегда определяет только место, отводимое на номер страницы, и эту команду можно переопределять в преамбуле. С другой стороны, параметр `\@tempdima` используется \LaTeX 'ом для самых разных целей (в основном — для временного хранения различных длин в процессе каких-то вычислений), и он может измениться в процессе выполнения очень многих \LaTeX 'овских команд. Поэтому присваивать ему какое-то значение в преамбуле совершенно бессмысленно — все равно после этого оно не раз изменится. Как мог заметить читатель, значение этому параметру присваивается в начале исполнения команды `\@chapter`, и именно это значение принимается в расчет в дальнейшем. Поэтому, если вы захотите отводить на номер главы, скажем, `2em` вместо `1.5em`, то вам придется переопределить в преамбуле команду `\@chapter`, заменив третью строку на

```
\@tempdima=1.5em
```

Нужда в таком переопределении `\@chapter` возникает, если мы переопределяем команду `\thechapter`, чтобы номер печатался римскими цифрами (как в книге, которую вы читаете). Далее, `#1` — это, как уже было сказано, номер и заглавие главы. Точнее говоря, на месте `#1` печатается²⁶ такой текст (будем считать, что глава называется «Все о слонах»):

```
\makebox[\@tempdima][1]{\thechapter} Все о слонах
```

Таким образом, отступ от левого поля до заглавия главы всегда равен `\@tempdima`; если номер занимает больше места, чем `\@tempdima`, то он наложится на заглавие.

Команда `\hfil` в десятой строке обеспечивает пробел между заглавием главы и номером страницы. Если вы хотите заполнить этот пробел лидерами, можете в определении `\@chapter` заменить `\hfil` на, скажем,

```
\leaders\hbox to .5em{\hss.\hss}\hfil
```

²⁶ Начиная с левого поля, невзирая на установленное значение `\leftskip`; именно для этих целей в определение включена команда `\hskip{-\leftskip}`.

(автор не гарантирует, что именно при таком выборе параметров лидеры будут выглядеть красиво).

Команды, определяющие вид записей в списке иллюстраций (соответствующих плавающим иллюстрациям) и списке таблиц (соответствующих плавающим таблицам) называются `\lof` и `\lot` соответственно и определяются в стандарте с помощью `\dottedtocline`.

До сих пор речь шла о сборке материала для оглавления, списка иллюстраций и т. п. Однако же и у самого оглавления есть заголовок, и его оформление тоже можно менять. Чтобы было понятно, как это делать, опишем, как определена команда `\tableofcontents` в стандартном стиле `article`:

```
\newcommand{\tableofcontents}{%
{\section*{\contentsname}\@starttoc{toc}}}
```

Здесь `\contentsname` — это уже знакомая нам команда, которую при работе с русскими текстами приходится переопределять (см. с. 155). Как видите, заголовок оглавления оформляется просто как заголовок ненумерованного раздела. Вы можете вместо этого оформить заголовок, скажем, с помощью `\subsection`, или еще каким-либо образом. Новой для вас будет команда `\@starttoc`. У этой команды предусмотрен один обязательный аргумент. Этим аргументом должен быть `toc` (для оглавления), либо `lot` или `lof` (для списка таблиц или иллюстраций, соответственно). Команда `\@starttoc` читает `toc-` (соответственно, `lot-` или `lof-`) файл и создает оглавление как таковое.

На самом деле в определении `\tableofcontents` присутствует еще команда, позволяющая задать текст для включения в колонтитулы (вспомним, что `\section*` сама по себе никакой информации для колонтитулов не дает). Мы не будем вдаваться в скучные подробности по поводу этой опущенной нами команды. Когда, по прочтении разд. 6, вы научитесь задавать такие команды, вы сможете соответствующим образом переопределить и `\tableofcontents`.

5. Перечни общего вида

Теперь мы, наконец, можем завершить наш рассказ о том, как менять стиль оформления перечней (см. разд. VII.2.5).

Начнем с важного предупреждения. Чтобы отойти от стандартного оформления перечней, необходимо, естественно, изменить значение каких-то из перечисляемых в этом разделе параметров. Однако же, если вы попробуете попросту присвоить этим параметрам новые значения в преамбуле, то можете с удивлением обнаружить, что действия это не возымело. Поэтому, если уж вы начали читать этот раздел, дочитайте

его, пожалуйста, до конца: там описаны специальные средства, которые надо применять, чтобы эти изменения подействовали.

5.1. Параметры, влияющие на оформление перечней

Для начала договоримся о терминологии. Каждый перечень \LaTeX рассматривает как состоящий из элементов (каждый элемент вводится, как мы помним, командой `\item`). В свою очередь, каждый элемент перечня может состоять из одного или нескольких абзацев. Наконец, у каждого элемента перечня есть свой заголовок — «горошина» на первом уровне окружения `itemize`, заданный вами заголовок в окружении `description`, и т. п.

Вооружившись этими терминами и имея в виду предупреждение, приступим к утомительному перечислению параметров. Все они — параметры со значением длины. Во-первых, параметры `\leftmargin` и `\rightmargin` задают, с каким отступом от левой (соответственно, правой) границы текста начинается (соответственно, заканчивается) текст элементов перечня (полиграфист сказал бы: насколько *втянуты* элементы перечня). Если перечень вложен в другой перечень, то `\leftmargin` и `\rightmargin` обозначают величину *втяжки* по отношению к объемлющему перечню.

Следующие два параметра влияют на размещение заголовков в перечне. Параметр `\labelsep` задает расстояние между правым краем заголовка и началом текста в элементе перечня, к которому относится этот заголовок, а параметр `\labelwidth` задает место по горизонтали, которое по умолчанию занимает заголовок. Точный смысл этих параметров следующий. При обработке перечня \LaTeX сначала пытается поместить заголовок в блок шириной `\labelwidth`. Если места хватает, то именно в такой блок он и помещается, причем выключенным вправо: правый край блока при этом находится на расстоянии `\labelsep` от начала текста, составляющего элемент перечня (так что его левый край будет на расстоянии

`\leftmargin - \labelwidth - \labelsep`

от левой границы основного текста или объемлющего перечня). Если же ширина заголовка больше, чем `\labelwidth`, то заголовок печатается как есть. Такое, например, регулярно случается при пользовании окружением `description`.

Мы не сказали еще об одном параметре, влияющем на размещение заголовков. Именно, если параметр `\itemindent` отличен от нуля, то каждый заголовок перечня будет дополнительно сдвинут на это расстояние вправо. Соответственно, при определении, на каком расстоянии начинается заголовок элемента

перечня, надо будет прибавить значение `\itemindent` к тому, что получается по формуле (*). По умолчанию значение этого параметра равно нулю.

Если элемент перечня состоит из нескольких абзацев, то по умолчанию во всех этих абзацах абзацный отступ будет отсутствовать. Можно, однако, при желании задать такой режим, что во всех, кроме первого, абзацах каждого элемента перечня будет присутствовать абзацный отступ. Для этого надо задать ненулевую величину этого отступа в параметре `\listparindent`. Кстати, значение этого параметра может быть и отрицательным (в этом случае эффект будет похож на тот, что достигается в обычном тексте установкой параметров `\hangindent` и `\hangafter`).

Параметры, о которых шла речь до сих пор, относились к размещению материала по горизонтали. Теперь займемся «вертикальными» параметрами. Сразу отметим, что все эти параметры являются «растяжимыми» длинами (с. 122), т. е. у них можно задавать `plus-` и `minus-` компоненты.

Первый (и основной) из этих параметров называется `\topsep`. Это величина дополнительного вертикального интервала, который делается перед перечнем и после него (в дополнение к `\parskip` — см. с. 124).

Если перед перечнем оставлена пустая строка (или имеется команда `\par`), то перед и после перечня устанавливается еще и вертикальный отступ, равный `\partopsep` (в дополнение к отступам, заданным параметрами `\parskip` и `\topsep`).

Далее, вертикальный отступ между абзацами внутри одного элемента задается параметром `\parsep` (а не `\parskip`, как в обычном тексте). Между различными же элементами перечня, в дополнение к `\parsep`, оставляется еще и вертикальный отступ `\itemsep`. Таким образом, если `\itemsep` отличен от нуля, как это и сделано в стандартных стилях, то различные элементы перечня будут более отделены друг от друга, чем абзацы внутри одного элемента перечня.

5.2. Окружения `list` и `trivlist`

Все L^AT_EX'овские перечни являются на самом деле частными случаями одной общей конструкции — окружения `list`. Рассмотрим, как это окружение работает.

Первое, что необходимо усвоить: окружение `list` имеет два обязательных аргумента. Поэтому общий вид окружения `list` в исходном тексте будет такой:

```
\begin{list}{заголовок_по_умолчанию}{команды}
```

.....
\end{list}

Аргументы окружения `list` означают следующее. «Заголовок по умолчанию» — это заголовок элемента перечня, печатающийся в том случае, когда этот элемент перечня вводится командой `\item` без аргумента. Пример:

И какой-то малыш показал ему шиш.

И какой-то барбос укусил его в нос. Нехороший барбос, невоспитанный!

```
\begin{list}{И какой-то}()
\item малыш показал ему шиш.
\item барбос укусил его в нос.
Нехороший барбос, невоспитанный!
\end{list}
```

Аргумент команды окружения `list` содержит те команды, которые будут исполнены после входа в перечень. Поэтому в нем можно задать команды, присваивающие новые значения параметрам оформления перечня. Кроме этого, во втором аргументе окружения `list` можно поместить команду `\usecounter`. Эта последняя требует одного обязательного аргумента — имени счетчика (счетчик должен быть определен). Если `\usecounter` присутствует во втором аргументе окружения `list`, то при входе в окружение значение счетчика, являющегося аргументом `\usecounter`, будет установлено в нуль, а каждая команда `\item` без аргумента будет увеличивать его на единицу с помощью `\refstepcounter` (так что на значения этого счетчика можно будет ссылаться с помощью `\label` и `\ref`). Вот пример с `\usecounter`:

Вот как выглядят первые буквы латинского алфавита:

A: Выглядит так же, как соответствующая русская буква, и читается так же.

B: Читается не так, как похожая на нее русская буква.

C: И с ней та же история.

Вот как выглядят первые буквы латинского алфавита:

```
\begin{list}{\Alph{tmp}:}%
\usecounter{tmp}
\item Выглядит так же, как соответствующая русская буква, и читается так же.
\item Читается не так, как похожая на нее русская буква.
\item И с ней та же история.
\end{list}
```

Чтобы заголовки элементов перечня выравнивались по левому краю, а не по правому, то можно завершить «заголовок по умолчанию» командой `\hfill`; чтобы по левому краю выравнивались заголовки, заданные в явном виде в необязательном аргументе команды `\item`, нужно завершить командой `\hfill` этот необязательный аргумент.

Окружением `list` разумно пользоваться не непосредственно, как в приведенных примерах, а для определения нового окружения с помощью `\newenvironment`. Вот, например, как в стандарте определяется окружение `quote`:

```
\newenvironment{\quote}{%
{\begin{list}{}{\rightmargin=\leftmargin}\item[]}}%
{\end{list}}
```

Команда `\item` с пустым аргументом необходима, поскольку до команды `\item` в перечне не должно быть никакого текста (см. с. 129).

Наряду с окружением `list` в `LATEX`'е определен его важный частный случай — окружение `trivlist`. Его отличия от `list` таковы:

- Это окружение не требует аргументов (так же, как и все окружения для создания перечней, с которыми мы имели дело раньше).
- `\leftmargin`, `\labelwidth` и `\itemindent` для него всегда равны нулю (стало быть, текст печатается без втяжки); `\parsep` равен `\parskip`.
- Команда `\item`, употребленная внутри этого окружения, обязана иметь аргумент (хотя бы пустой).

Как видите, многие возможности перечней в этом окружении не работают, но сохраняются такие важные черты, как `\topsep` (дополнительный интервал перед и после) плюс обычное свойство первой строки после перечня: она делается без абзацного отступа тогда и только тогда, когда после окружения в исходном тексте не оставлено пустой строки. Окружение `trivlist` также применяют обычно не само по себе, а для определения новых окружений; при этом в «открывающие команды» `\newenvironment` добавляют команду `\item[]`, а внутри окружения `\item` вообще не используют. Иногда используют и `\item` с аргументом (пример тому вы увидите ниже, в разд. 8.1).

6. Колонтитулы

Страницу документа, подготовленного с помощью `LATEX`'а, можно рассматривать как состоящую из трех частей: тела страницы, верхнего

колонтитула и нижнего колонтитула²⁷. Мы знаем, что с помощью команды `\pagestyle` на оформление колонтитулов можно в какой-то мере влиять. Возможностей для этого, однако же, не слишком много. Сейчас мы увидим, как можно радикально менять вид колонтитулов.

За оформление верхних колонтитулов отвечают команды `\Oddhead` и `\Evenhead`. Точнее говоря, если стиль оформления документа «двустронний», то команда `\Oddhead` задает верхний колонтитул на страницах с нечетными номерами, а команда `\Evenhead` — на страницах с четными номерами. Если же стиль оформления документа «односторонний», то `\Oddhead` задает все верхние колонтитулы, а команда `\Evenhead` на оформление документа вообще не влияет. Аналогично обстоит дело с `\Oddfoot` и `\Evenfoot`, отвечающими за нижние колонтитулы. Все четыре названные команды получают некоторое определение в *LATEX'овском стандарте*, так что переопределять их надо с помощью `\renewcommand`.

Теперь обсудим, как вышеназванные команды влияют на оформление колонтитулов. Основной принцип таков. При оформлении страницы верхний колонтитул получается в результате исполнения команды

```
\hbox to \textwidth{\Evenhead}
```

(мы предположили, что стиль двусторонний и страница четная; в других случаях — с очевидными изменениями). Можно сказать, что в каждой из команд `\Evenhead` и ей подобных записан текст и *TeX'овские* команды, которые при верстке страницы будут подставлены в `\hbox to \textwidth`.

Пусть, например, мы готовим к изданию роман Л. Н. Толстого «Война и мир»; стиль документа выберем двусторонний. Предположим, что мы выбрали такое оформление:

- На левой странице разворота в верхнем колонтитуле написано имя автора, прижатое (выключенное) вправо.
- На правой странице разворота в верхнем колонтитуле написано название романа, прижатое (выключенное) влево.
- В нижних колонтитулах по центру расположен номер страницы, окруженный тире.

Тогда надо переопределить команды для колонтитулов так:

```
\renewcommand{\Evenhead}{\hfil Л. Н. ТОЛСТОЙ}
\renewcommand{\Oddhead}{ВОЙНА И МИР\hfil}
```

²⁷Эта терминология не совпадает с традиционной, но удобна для наших целей.

```
\renewcommand{\@evenfoot}{\hfil --- \thepage ---\hfil}
\renewcommand{\@oddfoot}{\hfil --- \thepage ---\hfil}
```

Итак, мы научились менять оформление колонтитулов, переопределяя команду `\@evenhead` и ей подобные. А теперь — важное предупреждение: команда `\pagestyle` также переопределяет команды типа `\@evenhead`; если вы проведете переопределения в преамбуле документа, то первой же командой `\pagestyle` ваши переопределения будут отменены. Стало быть, переопределения команд типа `\@evenhead` часто приходится давать не в преамбуле, а внутри документа; разумеется, перед такими переопределениями должна стоять команда `\makeatletter` (чтобы вы смогли работать с командами, имеющими `\` в имени), а после них — восстанавливающая *status quo* команда `\makeatother`. Сами переопределения, естественно, надо давать сразу после `\newpage` или `\clearpage` (если, конечно, вы хотите понимать, с какой страницы начнется новое оформление).

Если мы хотим, чтобы каких-то колонтитулов вообще не было, то надо переопределить соответствующую команду на «ничего не делать», например, так:

```
\renewcommand{\@oddfoot}{}
```

Прежде чем перейти к более сложным вещам, скажем еще о трех стилевых параметрах. Во-первых, интервалы между колонтитулами и текстом регулируются параметрами со значением длины `\headsep`, задающим интервал между верхним колонтитулом и текстом, и `\footskip`, задающим расстояние между базисной линией последней строки в теле страницы и базисной линией нижнего колонтитула. Во-вторых, существует параметр `\headheight`, задающий высоту верхних колонтитулов. Если сумма высоты и глубины заданного вами колонтитула будет превышать `\headheight`, то при трансляции вы получите сообщение

`Overfull \vbox occurred while \output was active.`

Приведем пример, когда надо учитывать `\headheight`. Пусть мы хотим в том же издании «Войны и мира» отделять верхние колонтитулы от текста линейками. Разумный способ это сделать — передать в `\vbox` то `\textwidth` уже готовый блок шириной `\textwidth`, содержащий как текст колонтитула, так и линейку. Так как линейку под текстом удобно проводить в вертикальном режиме с помощью команды `\rule`, в голову приходит вот что (здесь и далее мы для краткости приводим только определение `\@evenhead`):

```
\renewcommand{\@evenhead}{%
{\vbox{\hbox to\textwidth{\hfil Л. Н. ТОЛСТОЙ}\rule{0pt}{1ex}}}}
```

У такого определения есть, однако, два недостатка. Во-первых, так как линейки в вертикальном режиме добавляются «впритык» к предшествующему блоку, у нас нет гарантии, что линейка не подойдет к тексту слишком близко; если бы к тому же текст в колонтитулах на разных страницах варьировался, как в дальнейших примерах, то может случиться и так, что две соседние линейки на развороте находятся на разной высоте (из-за того, что в колонтитуле на одной странице есть буквы вроде *у*, опускающиеся ниже базисной линии, а на другой — нет). Средство от этого недостатка — добавить `\strut` в наш блок:

```
\renewcommand{\@evenhead}{%
{\vbox{\hbox to\textwidth{\hfil \strut
Л. "Н. "ТОЛСТОЙ}\hrule}}}
```

Второй недостаток — это то, что к размеру блока с текстом добавится ширина линейки, в результате чего этот размер превысит `\headheight` и мы будем получать уйму сообщений об *Overfull*'е. Чтобы уйти от этого, надо еще чуть-чуть схитрить:

```
\renewcommand{\@evenhead}{%
{\raisebox{0pt}[\headheight][0pt]{% начало блока
\vbox{\hbox to\textwidth{\hfil \strut
Л. "Н. "ТОЛСТОЙ}\hrule}}% конец блока
}% конец макроопределения}
```

Понятно ли, что происходит? С помощью `\raisebox` мы заставляем TeX считать, что блок имеет высоту `\headheight` (нам даже незачем виникать, чему она фактически равна) и нулевую глубину (чтобы в сумме получилось то, что надо). Теперь ни о каких переполнениях речи не будет; пробел между колонтитулом и текстом можно при желании изменить, изменив значение `\headsep`.

Для нижних колонтитулов параметра, аналогичного `\headheight`, нет, так что смело делайте их любой высоты.

Кстати говоря, команду `\@evenhead` в этом примере можно было бы переопределить более простым образом, без `\vbox`, с использованием команды `\underline`. Наш способ, однако, более гибок: например, мы можем регулировать толщину линейки, чего с `\underline` не добьешься.

Итак, мы научились создавать собственные колонтитулы. Однако в L^AT_EX'овском стандарте в колонтитулы обычно помещается не только номер страницы, но и, например, номер и заглавие текущего раздела. Давайте научимся делать и это.

Чтобы передать в колонтитул какую-то информацию из текста, в L^AT_EX'е используются команды `\markboth` и `\markright`. Разберем, как

они работают. Команда `\markboth`, требующая двух обязательных аргументов, заносит в текст пару «пометок» — два фрагмента текста (возможно, с ТeX'овскими командами). Например, если мы скажем (где-нибудь между абзацами)

```
\markboth{Кот}{Пес}
```

то поместим между этими абзацами пару пометок: «левую пометку» Кот и «правую пометку» Пес. Самы по себе эти пометки никак не отражаются на печати. Однако же в определениях команд `\@oddhead` и ей подобных мы можем на эти пометки ссылаться. Именно, в этих определениях команда `\leftmark` дает левую пометку, а команда `\rightmark` — правую пометку. Например, если мы переопределим верхние колонтитулы как

```
\renewcommand{\@evenhead}{\leftmark\hfil}
\renewcommand{\@oddhead}{\hfil\rightmark}
```

то, начиная с той страницы, на которую попали наши пометки, в левом верхнем колонтитуле будет стоять выключенное влево слово «Кот», а в правом — выключенное вправо слово «Пес»²⁸. Кстати, если никаких пометок в тексте нет, то как `\leftmark`, так и `\rightmark` дают «пустой» текст.

До сих пор мы молчаливо предполагали, что на каждой странице присутствует только одна пара пометок. Что будет, если таких пар пометок попадет на страницу несколько? Ответ: команда `\leftmark` в этом случае означает левую пометку из *самой верхней* пары пометок, попавших на страницу, а команда `\rightmark` означает правую пометку из *самой нижней* пары пометок, попавших на данную страницу. С другой стороны, если на страницу вообще ни одна пара пометок не попала, то `\leftmark` и `\rightmark` означают соответственно левую и правую пометки из последней пары пометок, встретившихся до этого.

Поясним сказанное примером. Пусть пометки

```
\markboth{x}{y} и
\markboth{z}{t}
```

попали (в указанном порядке) на страницу 1, на страницу 2 вообще никаких пометок не попало, на страницу 3 попали целые три пары пометок:

²⁸ Точнее говоря, так будет, если в тексте нет команд типа `\section`: эти команды, как мы увидим ниже, автоматически вставляют в текст свои пометки, что усложняет картину.

```
\markboth{u}{v},
\markboth{a}{b} и
\markboth{m}{n}
```

а на страницах 4 и дальнейших никаких новых пометок не появлялось. Тогда значения команд `\leftmark` и `\rightmark` в процессе обработки этих страниц были таковы:

Страница	<code>\leftmark</code>	<code>\rightmark</code>
1	<i>x</i>	<i>t</i>
2	<i>z</i>	<i>t</i>
3	<i>u</i>	<i>n</i>
4 и далее	<i>m</i>	<i>n</i>

Наряду с командой `\markboth`, которая ставит в тексте новую пару пометок, есть еще и команда `\markright` с одним аргументом, которая ставит в тексте пару пометок таким образом: левый элемент в этой паре — такой же, как был в предыдущей паре пометок, а правый — тот, что задан в аргументе команды `\markright`. Например, если сначала идет команда

```
\markboth{Кот}{Лес}
```

а затем команда

```
\markright{Собака}
```

(и в промежутке между этими командами никаких других пометок в текст не вносится), то это равносильно тому, как если бы вторая из этих команд была

```
\markboth{Кот}{Собака}
```

Выше мы уже отмечали, что команды типа `\section` ставят в тексте пометки автоматически. Понять, как это делается и как можно влиять на этот процесс, проще всего на примере. Сделаем такие предположения о стиле документа: основной стиль — `article`, разделы и подразделы нумеруются (иными словами, значение счетчика `secnumdepth` больше единицы), действует стилевая опция `twoside` и в преамбуле была дана команда `\pagestyle{headings}`.

Во-первых, отметим, что в этом случае пометки в текст вставляют только команды `\section` и `\subsection`. При этом команда `\section` ставит пометки, автоматически выполняя команду `\sectionmark`, имеющую один обязательный аргумент — заглавие раздела (если команда `\section` была дана без обязательного аргумента) или вариант заглавия, заданный в необязательном аргументе команды `\section` (если

таковой присутствует). Вот как определена `\sectionmark` (на место #1 будет подставляться вариант заглавия, идущий в колонтитулы):

```
\newcommand{\sectionmark}[1]{\markboth{%
\uppercase{\thesection\hspace{1em}#1}}% левая пометка
{}% правая пометка (она пуста)
}% конец макроопределения
```

Здесь используется ТЕХ'овская команда `\uppercase`, которую мы ранее не рассматривали. Не вдаваясь в подробности, скажем, что эта команда переводит все буквы в тексте²⁹, попавшем в ее аргумент, из строчных в прописные. Коль о том зашла речь, отметим, что есть еще и команда `\lowercase`, которая наоборот переводит все буквы в тексте, попавшем в ее аргумент, из прописных в строчные. Не пытайтесь, пожалуйста, использовать `\uppercase` и `\lowercase` вне аргументов `\markboth` или `\markright`, иначе вас подстерегают неприятные сюрпризы. Если вы не хотите, чтобы в колонтитулах строчные буквы заменялись на прописные, можно при переопределении просто опустить команду `\uppercase` (так и было сделано при оформлении книги, которую вы читаете).

Команда `\subsectionmark`, определяющая вид пометок, автоматически вставляемых в текст командой `\subsection`, определена следующим образом:

```
\newcommand{\subsectionmark}[1]{\markright
{\thesubsection\hspace{1em}#1}}
```

Здесь также аргумент #1 — это заглавие подраздела (точнее, его вариант для колонтитулов).

Итак, в рассматриваемом нами случае команда `\section` вносит в текст следующую пару пометок: заглавие раздела, в котором все буквы заменены на прописные, в качестве левой пометки, и пустой текст в качестве правой пометки. Команда же `\subsection` вносит в текст пару пометок, в которой левая пометка такая же, как в предыдущей паре, а правая — заглавие подраздела (точнее, его вариант для колонтитулов), причем на сей раз «в натуральном виде», без замены строчных букв на прописные. Как уже отмечалось выше, команды `\subsubsection` и более мелкие никаких пометок в текст не вносят.

Посмотрим, как в этом стиле используются пометки. Колонтитулы на четных страницах в этом случае определены так:

```
\newcommand{\@evenhead}{\rm\thepage\hfil\sl\leftmark}
```

²⁹ Но не в именах команд.

Тем самым на страницах с четными номерами (они будут левыми на развороте) колонтитул будет выглядеть следующим образом: выключенный влево номер страницы (прямым шрифтом), и заглавие текущего раздела (наклонным шрифтом, прописными буквами) — выключенное вправо³⁰: ведь никаких других левых пометок в тексте нет!

Команда `\Ooddhead`, отвечающая за верхние колонтитулы на страницах с нечетными номерами, определена в стиле `article` (при условии, что была команда `\pagestyle{headings}`) так:

```
\newcommand{\Ooddhead}{\rlap{\rightmark}\hfil \rm\thepage}
```

Стало быть, верхние колонтитулы к нечетным страницам выглядят так: название текущего подраздела наклонным шрифтом — выключенное влево, номер страницы прямым шрифтом — выключенный вправо. Впрочем, если в текущем разделе команда `\subsection` еще не было, то вместо заглавия будет пустое место, так как `\rightmark` будет пуста (определение команды `\sectionmark`, данное выше, показывает, что при исполнении команды `\section` в текст вносится пустая правая пометка, и пустой она остается до первой команды `\subsection`).

Приведем пример переопределения команд наподобие `\sectionmark`. Если нам не нравится, что в правом колонтитуле иногда бывает пустое место, то можно попросить команду `\section` вносить в текст непустую правую пометку — то же заглавие раздела. Для этого напишем в преамбуле так:

```
\newcommand{\sectionmark}[1]{\markboth
{\uppercase{\thesection\hspace{1em}\#1}}% левая пометка
{\uppercase{\thesection\hspace{1em}\#1}}% правая пометка
}% конец макроопределения
```

Теперь, если в разделе нет подразделов, то на правых страницах в верхнем колонтитуле будет также печататься заглавие текущего раздела. Оформление колонтитулов при этом будет стандартным. Если вы хотите еще и отойти от этого стандарта, то надо будет, вместо использования команды `\pagestyle`, напрямую переопределить команды типа `\Ooddhead` и `\Oevenhead`; надо думать, теперь вы найдете, как можно распорядиться в этих определениях `\leftmark`'ом и `\rightmark`'ом.

Если в документе присутствует команда `\pagestyle` (с каким бы то ни было аргументом), то она отменит ваши переопределения команд типа `\sectionmark`. Выход такой же, как при переопределении команд

³⁰Точнее говоря, это будет заглавие либо текущего раздела, либо первого из разделов, начинающихся на этой странице — см. выше обсуждение `\leftmark` и `\rightmark`.

типа `\@evenhead`: переопределять `\sectionmark` и иже с ней не в преамбуле, а внутри документа, после `\newpage` или `\cleargpage`.

Если вы вообще не хотите, чтобы при исполнении, скажем, команды `\subsubsection` в текст вносились какие-либо пометки, то можно «отключить» команду `\subsubsectionmark`, переопределив ее на «ничего не делать»:

```
\renewcommand{\subsubsectionmark}{[1]{}}
```

Бывает и так, что вас устраивает стандартный стиль оформления колонтитулов, но при этом не устраивает, что именно в эти колонтитулы автоматически записывается. Например, у ваших разделов длинные заглавия, и вы при этом хотите, чтобы они в несокращенном виде попали в оглавление, а сокращенные варианты заглавий пошли только в колонтитулы. Команда `\section` с необязательным аргументом тут не спасет, так как этот необязательный аргумент запишется и в оглавление тоже. Вот бы задать информацию для колонтитулов вручную, без того, чтобы это автоматически делали команды типа `\section!` Можно, конечно, переопределить на «ничего не делать» все команды типа `\sectionmark`, как в приведенном выше примере, но `ЛАTeX` предоставляет вам более простой способ. Если написать в преамбуле

```
\pagestyle{myheadings}
```

то все `ЛАTeX`'овские команды для создания разделов не будут вносить пометок в текст, а оформление колонтитулов будет стандартным. Тогда вы сможете, например, написать

```
\section{О некоторых специальных свойствах  
подмножеств пустых множеств, не рассматривавшихся  
в предыдущих разделах статьи}%  
\markboth{\thesection\hspace{1em}Подмножества}{}
```

и в оглавлении будет заголовок целиком, а в колонтитуле — всего лишь слово «Подмножества» (мы подразумеваем, что стиль все тот же, только аргументом команды `\pagestyle` был `myheadings` вместо `headings`). Мы поместили команду, вносящую в текст пометки, *после* команды `\section`, чтобы номер раздела, генерируемый командой `\thesection`, был правильным. Кроме того, мы убрали с помощью знака процента пробел (конец строки) между командами `\section` и `\markboth`, чтобы пометки с гарантией попали на ту же страницу, что и заголовок раздела.

Другие команды, отвечающие за автоматическую расстановку пометок в тексте, устроены аналогичным образом: команда `\chapter` ставит пометки с помощью команды `\chaptermark`, команда `\section` —

с помощью команды `\sectionmark`, и т. д. — вообще, если команда, генерирующая раздел документа, определена с помощью `\@startsection` с первым аргументом `abcd`, то она будет ставить пометки с помощью команды `\abcdmark`. Все эти команды автоматически выполняются в процессе выполнения соответствующей команды, генерирующей раздел. Они должны иметь один обязательный аргумент, в качестве которого им передается заглавие раздела (точнее, вариант этого заглавия, предназначенный для колонтитулов и оглавления). Варианты «со звездочкой» команд, генерирующих разделы документа, никаких пометок в текст не вносят (как и следовало ожидать).

Если в аргументе команды `\markboth` или `\markright` присутствует не только текст, но и какие-то команды, то в пометки будут записаны не буквально эти команды, но их значение на тот момент, когда пометки вносятся в текст. Например, если в аргументе `\markboth` присутствует команда `\thesection`, то в пометку реально запишется (и, соответственно, прочтется из `\leftmark` или `\rightmark`) номер раздела. Если же необходимо, чтобы какая-то команда записалась в пометку в том же виде, в каком она была задана в аргументе `\markboth` или `\markright`, то надо «защитить» ее, поставив перед ней `\protect`.

Еще один случай, когда пометки в текст вносятся \LaTeX 'ом автоматически, возникает при оформлении таких фрагментов документа, как оглавление, список иллюстраций или таблиц, список литературы и предметный указатель. Как именно они вносятся, зависит от основного стиля и от того, пользовались ли мы (и как пользовались) командой `\pagestyle`. Именно, если основной стиль — `article`, то никаких пометок при оформлении оглавления и т. п. в текст автоматически вноситься не будет; точно так же не будет этих пометок при любом основном стиле после того, как выполнится команда `\pagestyle` с аргументом `empty`, `plain` или `myheadings`. С другой стороны, если основной стиль — `report` или `book`, то, например, при исполнении команды `\tableofcontents` автоматически выполняется и команда

```
\markboth{\contentsname}
```

(с очевидными изменениями для списка иллюстраций и т. п.). Та же ситуация возникнет и после исполнения команды `\pagestyle` с аргументом `headings` (даже тогда, когда основной стиль — `article`).

Итак, вы теперь имеете широкие возможности для создания собственного оформления колонтитулов. Подведем итоги. Во-первых, можно написать

```
\pagestyle{myheadings}
```

При этом внешний вид колонтитулов будет стандартный, но материал для колонтитулов вы будете поставлять вручную с помощью команд `\markboth` и/или `\markright`. Во-вторых, можно переопределять команды типа `\sectionmark`: при этом внешний вид колонтитулов будет по-прежнему стандартный, но материал для помещения в колонтитул будет отбираться автоматически по схеме, отличной от стандартной. В-третьих, можно переопределять команды типа `\@oddhead`: при этом вы меняете как стиль оформления колонтитулов, так и способ обращения с пометками, вносимыми в текст (в частности, если в ваших определениях команда типа `\@oddhead` не участвуют `\leftmark` и `\rightmark`, то эти пометки будут вообще проигнорированы). Наконец, можно объединить второй и третий подходы. При этом вы самостоятельно разрабатываете как внешний вид колонтитулов, так и то, какая информация из текста и как будет в них отражена.

7. Плавающие объекты

В этом разделе мы выполним обещание, данное на с. 169, и обсудим параметры, которыми руководствуется \LaTeX при размещении плавающих иллюстраций и таблиц (короче: плавающих объектов) на странице.

Сразу же отметим, что все эти параметры относятся в равной мере к плавающим иллюстрациям и плавающим таблицам (а также и к плавающим объектам других типов, буде вы их определите — см. ниже в этом разделе), и любое изменение этих параметров также затрагивает плавающие объекты всех типов.

Часть параметров, отвечающих за плавающие объекты, представляют собой \LaTeX 'овские счетчики, которым можно присваивать новые значения командой `\setcounter`:

topnumber Максимальное количество плавающих объектов, которое разрешается разместить вверху страницы (при двухколонном наборе и иллюстрациях/таблицах шириной в колонку — вверху колонки). Значение по умолчанию: 2.

bottomnumber Максимальное количество плавающих объектов, которое разрешается разместить внизу страницы (при двухколонном наборе и иллюстрациях/таблицах шириной в колонку — внизу колонки). Значение по умолчанию: 1.

totalnumber Максимальное количество плавающих объектов, которое разрешается разместить на странице (при двухколонном наборе и иллюстрациях/таблицах шириной в колонку — в колонке). Значение по умолчанию: 3.

dbltopnumber При наборе в две колонки: максимальное количество плавающих объектов шириной во всю страницу, которое разрешается разместить вверху страницы. Значение по умолчанию: 2.

Значение счетчика **totalnumber** не влияет на количество иллюстраций и/или таблиц на странице, специально для них предназначенной (таковая, как мы помним, выделяется, если в необязательном аргументе окружения **figure** или **table** присутствует буква **r**).

Вторая группа параметров регулирует уже не количество плавающих иллюстраций на странице, а величину места, ими занимаемого. Все эти параметры являются командами наподобие **\arraystretch** или **\baselinestretch**; чтобы менять значения параметров, надо их переопределять с помощью команды **\renewcommand**. Эти параметры таковы:

\topfraction Максимальная доля страницы, которую могут занимать плавающие объекты, размещаемые вверху страницы. Значение по умолчанию: 0.7. Это означает, что плавающие иллюстрации и таблицы, размещаемые вверху страницы, могут занимать не более 70% страницы по высоте. Если мы хотим уменьшить эту долю, скажем, до 50%, надо написать в преамбуле

```
\renewcommand{\topfraction}{0.5}
```

Ниже слова «доля страницы» также всюду означают «доля страницы по высоте».

\bottomfraction Максимальная доля страницы, которую могут занимать плавающие объекты внизу страницы. Значение по умолчанию: 0.3.

\textfraction Минимальная доля страницы, которую должен занимать текст, а не плавающие объекты (к страницам, создаваемым L^AT_EX'ом специально для размещения плавающих объектов при обработке необязательного аргумента **r**, это не относится). Значение по умолчанию: 0.2.

\floatpagefraction Этот параметр, напротив, относится именно к страницам, которые L^AT_EX создает при обработке необязательного аргумента **r**. Он указывает минимально возможную долю такой страницы, которую могут занимать размещаемые на ней плавающие иллюстрации и таблицы. Значение по умолчанию: 0.5 (стало быть, специальная страница для плавающих иллюстраций, которую вы требуете с помощью необязательного аргумента **r**

к окружению `figure`, не будет создана, пока эти иллюстрации занимают менее 50% высоты страницы текста).

Все вышеуказанные параметры применимы также к двухколонному набору и плавающим объектам шириной в колонку. Если же набор — в две колонки, а ширина плавающего объекта — целая страница, то:

`\dbltopfraction` То же, что `\topfraction`, применительно к иллюстрациям (таблицам) шириной в целую страницу при двухколонном наборе. Значение по умолчанию: 0.7.

`\dblfloatpagefraction` То же, что `\floatpagefraction`, применительно к иллюстрациям (таблицам) шириной в целую страницу при двухколонном наборе. Значение по умолчанию: 0.5.

При вычислении, какую часть страницы занимает плавающий объект, учитывается не только размер собственно иллюстрации или таблицы, но и величина вертикальных отступов («отбивок»), отделяющих этот объект от остального текста. Эти отступы задаются следующей группой параметров. Все эти параметры суть длины, присваивать им новое значение надо с помощью `\setlength`:

`\textfloatsep` Отступ между текстом и иллюстрациями (таблицами), размещаемыми вверху или внизу страницы.

`\floatsep` Отступ между двумя иллюстрациями (таблицами).

`\intextsep` Отступ между текстом и иллюстрациями (таблицами), размещаемыми посреди страницы (при обработке необязательного аргумента `h`).

Как водится, три перечисленных параметра применимы и к иллюстрациям (таблицам) шириной в колонку при двухколонном наборе. А вот как обозначаются соответствующие параметры, если набор в две колонки, а иллюстрация или таблица занимает по ширине всю страницу:

`\dbltextfloatsep` При двухколонном наборе — отступ между текстом и иллюстрацией (таблицей), занимающей всю страницу по ширине.

`\dblfloatsep` При двухколонном наборе — отступ между двумя иллюстрациями (таблицами), занимающими всю страницу по ширине.

На случай, если вы решитесь менять эти параметры с помощью команды `\setlength`, скажем для ориентировки, что при пользовании шрифтом кегль 11 значения `\floatsep`, `\intextsep` и `\dblfloatsep`

равны `12pt plus 2pt minus 2pt`, а значения `\textfloatsep` и `\dbltextfloatsep` равны `20pt plus 2pt minus 4pt`.

По умолчанию плавающие объекты не отделены от текста ничем, кроме вышеперечисленных отбивок. \LaTeX , однако же, предоставляет вам возможность сделать так, чтобы иллюстрации (таблицы) отделялись от текста как-то иначе (линейками, например). Для этих целей предусмотрены следующие три команды:

\topfigrule Разделитель между плавающим объектом, размещаемым вверху страницы (колонки) и остальным текстом.

\botfigrule Разделитель между текстом и размещаемым внизу страницы (колонки) плавающим объектом.

Эти две команды относились и к двухколонному набору, при условии, что иллюстрации (таблицы) занимают по ширине одну колонку (иными словами, если используются окружения `figure` или `table` без звездочек). А если используются варианты этих окружений со звездочками, то:

\dblfigrule То же, что `\topfigrule`, для случая, когда набор двухколонный, а плавающий объект занимает по ширине всю страницу.

По умолчанию указанные три команды ничего не делают. Если вы хотите задать явные разделители между текстом и иллюстрациями (таблицами), то эти команды надо определить (скажем, в преамбуле документа) с помощью `\newcommand`³¹. Определять эти команды нужно не произвольным образом: чтобы они правильно стыковались с \LaTeX ’овскими алгоритмами размещения плавающих объектов, нужно иметь в виду следующее:

- 1) каждая из этих команд будет вызываться в те моменты, когда \TeX находится в вертикальном режиме;
- 2) по окончании работы каждой из этих команд \TeX должен снова оказаться в вертикальном режиме;
- 3) текст, генерируемый каждой из этих команд, не должен, с точки зрения \TeX ’а, занимать места по вертикали.

Поэтому будет неправильно, если вы, решив отделять текст от иллюстраций линейкой, определите `\botfigrule` просто как `\hrule`: первое и второе условия при этом выполнены будут, а вот третье — нет, в результате чего \LaTeX сбьется со счета при решении вопроса о размещении иллюстраций. Формально правильным было бы такое решение:

³¹ Именно так, а не переопределить с помощью `\renewcommand!` К сожалению, в этом месте имеется несогласованность ...

```
\newcommand{\botfigrule}{\hrule\vspace{-0.4pt}}
```

(вспомним, что линейка, генерируемая командой `\hrule`, имеет по умолчанию толщину 0.4 пункта). Впрочем, формальной правильности мало: если вы опробуете такое определение на практике, то увидите, что линейка вплотную прилегает к иллюстрации, что никуда не годится. Правильно действовать так:

```
\newcommand{\botfigrule}{\vspace{-3pt}\hrule\vspace{2.6pt}}
```

Теперь мы проводим линейку не прямо по верхней кромке иллюстрации, а на три пункта выше; заключительное `\vspace{2.6pt}` нужно для того, чтобы в сумме получилось нулевое вертикальное смещение.

Будем надеяться, что теперь вы сможете разобраться с `\topfigrule` и `\dblfigrule`.

В заключение отметим, что разделители, определяемые `\topfigrule` и ей подобными командами, не обязаны быть именно линейками: необходимо только при их определении учитывать три перечисленных выше обстоятельства.

8. Разное

8.1. Теоремы, выключные формулы

Чтобы изменить оформление «теорем» (окружений, определяемых с помощью `\newtheorem`), надо переопределить команды `\begin{theorem}` и `\Opargbegintheorem` (первая из них отвечает за оформление «теорем» без обязательного аргумента, вторая — за оформление «теорем» с обязательным аргументом). Первая из них определена так:

```
\newcommand{\begintheorem}[2]{\begin{trivlist}\it
\item[\hspace{\labelsep}\bf #1\ #2]}
```

Здесь аргумент `#1` означает название «теоремы» (например, «Теорема», «Предложение», «Лемма» ... — в команде `\newtheorem` это слово являлось вторым обязательным аргументом), а аргумент `#2` означает номер «теоремы». Если мы, например, хотим, чтобы после номера «теорем» стояла точка, нам достаточно переопределить эту команду, добавив точку после `#2`. Что делать для того, чтобы сменить шрифт, которым печатаются номер или текст «теорем», также достаточно ясно.

Команда `\Opargbegintheorem` определяется так:

```
\newcommand{\Opargbegintheorem}[3]{\begin{trivlist}
\it\item[\hspace{\labelsep}\bf #1\ #2\ (#3)]}
```

Здесь #1 и #2 по-прежнему означают название и номер теоремы, а #3 — необязательный аргумент «теоремы» (обычно в качестве такового задается имя ученого, которому приписывается данная теорема).

Если оформление, задаваемое окружением `trivlist`, вас не устраивает, то можно переопределить две вышеуказанные команды более радикально. Общий принцип таков. Перед текстом «теоремы», не имеющей необязательного аргумента, исполняется команда `\begin{theorem}`; у этой команды должно быть два аргумента, причем первый из них — название «теоремы», а второй — ее номер. Если «теорема» имеет необязательный аргумент, то вместо `\begin{theorem}` перед ее текстом исполняется команда `\opargbegin{theorem}`, имеющая три аргумента: первые два — такие же, как у `\begin{theorem}`, и третий — необязательный аргумент «теоремы» (имя первооткрывателя). Наконец, после текста «теоремы» исполняется команда `\end{theorem}`, которая изначально определена очень просто:

```
\newcommand{\endtheorem}{\end{trivlist}}
```

В принципе можно переопределить все три эти команды, чтобы получить свое оформление «теорем» (например, в духе наших макросов для автоматической нумерации задач из гл. VII). Только следите, чтобы переопределения всех трех команд были согласованы друг с другом: если, например, вы изгоните `\begin{trivlist}` из определения `\begin{theorem}`, но при этом оставите команду `\endtheorem` в неприкосновенности, то на каждой «теореме» *Л^AT_EX* будет сообщать вам об ошибке (отсутствие баланса `\begin{`ов` и `\end{`ов}`).

- ε При пользовании *AMS-L^AT_EX*овскими классами документов (приложение Е) «теоремы» определяются иначе. Вряд ли, впрочем, вы сочтете нужным переделывать стандарт Американского математического общества.

Теперь скажем кое-что про стиль оформления номеров выключочных формул, заданных в виде окружения `equation`. Как вам уже известно, можно переопределить команду `\theequation` или (с помощью команды `\addtoreset`) изменить подчиненность счетчика `equation`; при этом изменится оформление самих номеров формул. Кроме этого, можно, в принципе, изменить то, что печатается возле этих номеров. Для этого следует переопределить команду `\eqnnum`. Изначально она определена так:

```
\newcommand{\eqnnum}{(\theequation)}
```

При желании можно заменить тут круглые скобки на что-нибудь другое. Имейте в виду, что номер выключной формулы обрабатывается

TeX'ом «в математическом режиме», как формула (латинские буквы по умолчанию набираются «математическим курсивом», и т п.).

8.2. Сноски

Стиль оформления сносок зависит от многих вещей. Начнем с пробела между страницей и сносками. Чтобы его изменить, надо применить команду `\setlength` с необычным первым аргументом. Вот, например, как выглядит команда, устанавливающая стандартную для L^AT_EX'a величину этого пробела:

```
\setlength{\skip\footins}{12pt plus 4pt minus 4pt}
```

Почему в первом аргументе `\setlength` целых две команды и что они означают, объяснить в рамках этой книги невозможно, так что воспринимайте этот рецепт для установки пробела между текстом и сносками чисто догматически (самые любознательные могут попытаться узнать всю правду из пятнадцатой главы книги [3]).

Далее, сноски обычно отделяются от текста не только пробелом, но и линейкой. Чтобы задать вид этой линейки, отличный от стандартного, либо задать какой-то другой разделитель, надо переопределить команду `\footnoterule`, которая в стандарте определена так:

```
\newcommand{\footnoterule}{\vspace*{-3pt}
  \hrule width .4\columnwidth
  \vspace*{2.6pt}}
```

К этому макроопределению необходим комментарий: непонятно, зачем нужны команды `\vspace`. Дело в том, что «текст», генерируемый командой `\footnoterule`, не должен, с точки зрения TeX'a, занимать места по вертикали (фактически он располагается внутри пробела между текстом и сносками, о котором шла речь выше). Поэтому мы сначала отступаем на 3 пункта вверх, затем печатаем линейку (вспомним, что по умолчанию линейка, генерируемая командой `\hrule`, имеет ширину 0.4 пункта), и затем спускаемся на 2.6 пункта вниз. В итоге получается, что линейка напечаталась, и места по вертикали мы не занимаем, поскольку $-3 + 0.4 + 2.6 = 0$.

Осталось еще неясным, зачем нужен `\vspace*{-3pt}`: не проще ли обойтись без этой команды, а после `\hrule` сказать `\vspace*{-0.4pt}`? Ответ: в этом случае линейка напечаталась бы вплотную к сноске. Ср. с. 304.

Если вы хотите изменить ширину или толщину линейки, команду `\footnoterule` можно переопределить; только не забудьте проследить, чтобы отрицательный `\vspace` скомпенсировал толщину линейки!

Можно, собственно говоря, сделать так, чтобы этой линейки вообще не было, сказав

```
\renewcommand{\footnoterule}{}
```

(уж тут-то места по вертикали мы не заемем!). Если вам вдруг понадобится задать совсем иной разделитель между сносками и текстом, можете переопределить команду `\footnoterule` принципиально по-иному. В этом случае необходимо знать следующее:

- 1) команда `\footnoterule` будет вызываться в те моменты, когда \TeX находится в вертикальном режиме;
- 2) по окончании работы команды `\footnoterule` \TeX должен снова оказаться в вертикальном режиме;
- 3) текст, генерируемый командой `\footnoterule`, не должен, с точки зрения $\text{\TeX}'a$, занимать места по вертикали.

Следующий параметр, от которого зависит оформление сносок, — это параметр со значением длины `\footnotesep`. Он означает следующее. В начале каждой сноски, для того чтобы линейка, отделяющая сноски от текста, не подходила к тексту слишком близко, вставляется невидимая линейка нулевой ширины наподобие `\strut` (см. разд. III.10.3). Так вот, `\footnotesep` задает высоту этой линейки.

За вид номеров сносок в тексте отвечает команда `\@makefnmark`. По умолчанию она определена следующим образом:

```
\newcommand{\@makefnmark}{\hbox{\mathsurround=0pt
$^{\@thefnmark}$}}
```

Здесь на место команды `\@thefnmark` при выполнении будет подставлен номер сноски (или то, что его заменяет, если мы пользовались командой `\footnotemark`). Обратите внимание, что номер сноски оформлен как верхний индекс в математической формуле — именно благодаря этому номера сносок печатаются над строкой. По этой же причине внутри группы, являющей собой аргумент команды `\hbox`, устанавливается в нуль параметр `\mathsurround` — иначе, если вы установили для него ненулевое значение, номер сноски будет окружен лишними пробелами.

И, наконец, самое главное — команда, генерирующая собственно текст сноски. Она называется `\@makefntext`. Вот ее стандартное определение, в котором аргумент `#1` обозначает текст сноски, а команда `\@thefnmark` означает то же, что и выше:

```
\newcommand{\@makefntext}[1]{\parindent=1em\noindent
\hbox to 1.8em{\hss$^{\@thefnmark}$#1}}
```

При переопределении этой команды следует иметь в виду, что она будет выполняться внутри аргумента команды `\parbox` с длиной строки, равной ширине колонки текста; в приведенном выше определении применена команда `\noindent`, чтобы подавить абзацный отступ в первом абзаце сноски, в котором будет печататься ее номер.

Переопределяя `\@makefntext` таким образом, чтобы изменилось оформление номера сноски, например, чтобы он печатался так¹⁰), не забудьте изменить оформление номера сносков и в самом тексте, для чего надо согласованным образом переопределить команду `\@makefntmark`.

8.3. Список литературы

Если вам не нравится, что библиографические ссылки печатаются в квадратных скобках, то вы можете переопределить команды `\@cite` и `\@biblabel`. По умолчанию они определены так:

```
\newcommand{\@cite}[2]{[{#1}\if@tempswa , #2\fi}]}
\newcommand{\@biblabel}[1]{[#1]\hfill}
```

Мы не будем даже пытаться объяснить, что означают заковыристые команды в первом из этих макроопределений и как второе из них согласуется со всем остальным, а отметим только, что в «замещающем тексте» первого из этих определений можно заменить квадратные скобки на какие-нибудь другие (скажем, круглые или косые), — и тогда в соответствующих скобках будет печататься ссылка, генерируемая командой `\cite`; аналогично, если заменить скобки в «замещающем тексте» второго из этих определений, то в соответствующих скобках будет печататься номер источника в списке литературы. Будем надеяться, что для вас уже очевидно, что при переопределении этих команд надо писать `\renewcommand` вместо `\newcommand`.

8.4. Предметный указатель

Первое, что вы можете изменить в оформлении предметного указателя (окружение `theindex`), — это отступы, создаваемые командами `\item`, `\subitem` и т. д. Для изменения отступов «на первом уровне» (создаваемых командой `\item`) надо переопределить команду `\@idxitem` (но не сам `\item!`). Чтобы вам было от чего отталкиваться, посмотрите на стандартное определение этой команды. Оно очень простое:

```
\newcommand{\@idxitem}{\par\hangindent=40pt}
```

¹⁰) Нет, нет, это только для примера.

Для изменения отступов, создаваемых такими командами, как `\subitem` и `\subsubitem`, надо переопределить непосредственно эти команды. Их стандартные определения также бесхитростны:

```
\newcommand{\subitem}{\par\hangindent=40pt\hspace*{20pt}}
\newcommand{\subsubitem}{\par\hangindent=40pt\hspace*{30pt}}
```

Наконец, команда `\indexspace`, создающая в предметном указателе дополнительный вертикальный пробел, определяется в стандартных стилях так:

```
\newcommand{\indexspace}{\par\vspace{10pt plus 5pt minus 3pt}}
```

Иногда хочется изменить оформление предметного указателя более существенным образом. Например, вы можете захотеть, чтобы ссылка на предметный указатель присутствовала в оглавлении (в L^AT_EX'овском стандарте это не предусмотрено); возможно также, что вы захотите предпослать предметному указателю небольшое введение, набранное во всю ширину страницы. Чтобы добиться таких вещей, надо переопределить команду `\theindex`. Ее стандартное определение в стилях `book` и `report` выглядит так:

```
\newcommand{\theindex}{\@restonecoltrue
\if@twocolumn\@restonecolfalse\fi
\columnseprule=0pt \columnsep=35pt
\twocolumn[\@makeschapterhead{\indexname}]%
\@mkboth{\uppercase{\indexname}}{\uppercase{\indexname}}%
\thispagestyle{plain}\parindent=0pt
\setlength{\parskip}{0pt plus .3pt}%
\let\item=\@idxitem}
```

Первая, вторая и последняя строки этого определения содержат незнакомые вам команды; мы не будем пытаться объяснить, что они значат, а только скажем, что менять эти места в определении команды `\theindex` нельзя. Зато все остальное в этом определении должно быть понятно читателю, усвоившему основную часть нашей книги. Именно, в третьей строке задаются параметры двухколонного оформления в предметном указателе: там сказано, что колонки в окружении `theindex` не надо разделять линейкой и задается промежуток между двумя колонками (см. разд. IV.5.1). В пятой строке дана команда, задающая (неким экзотическим, не рассматривавшимся нами способом) материал для колонтитулов. Она либо вносит левую и правую пометки, совпадающие со стандартным заглавием предметного указателя, либо ничего не делает. Если уж вы переопределяете `\theindex`, то можете в этой строке

написать `\markboth` вместо `\emkboth` (если хотите, чтобы эти пометки были), или вообще убрать эту строку, чтобы пометок не было. В шестой строке обратите внимание на команду, устанавливающую нулевое значение абзацного отступа. Менять ее не надо, поскольку именно с таким значением абзацного отступа согласовано действие команд `\item`, `\subitem` и т. п. (если вы поняли, что такое `\hangindent`, то поймете, каким именно образом согласовано).

Наконец, в четвертой строке стоит команда `\twocolumn` с необязательным аргументом. Как объяснялось в разд. III.8.6, то, что стоит в необязательном аргументе, будет напечатано во всю ширину страницы. В стандартном определении тут стоит команда `\makeschapterhead`, создающая заголовок неуемерованной главы (см. разд. 3), но вы можете записать туда и любой текст, который хотите предпослать собственно предметному указателю. Правда, тут возникает один технический момент: введение к предметному указателю вы, видимо, захотите редактировать, и нехорошо для этого всякий раз лазать в преамбулу. Один из возможных выходов таков. Запишите в вашем определении команды `\theindex` четвертую строку так:

```
\twocolumn[\makeschapterhead{\indexname}\input{ukaz.tex}]
```

Здесь `ukaz.tex` — файл, в который вы запишете свое введение к предметному указателю.

Если вы хотите, чтобы предметный указатель был отражен в оглавлении, то в необязательный аргумент команды `\twocolumn` надо добавить команду `\addcontentsline`, например, в таком виде:

```
\addcontentsline{toc}{chapter}{\indexname}
```

Последнее, что нужно сказать про команду `\theindex`, это то, что в стиле `article` четвертая строка ее стандартного определения выглядит так:

```
\twocolumn[\section*\{\indexname}\%]
```

(поскольку в стиле `article` главы не определены).

Приложение А

О ТЕХ'овских шрифтах

Большая часть шрифтов, используемых **ЛАТ $\mathrm{\TeX}$ 'ом**, была создана самим Дональдом Кнутом (в соавторстве с Германом Цапфом) с помощью им же написанной программы **METAFONT**. Эти шрифты называются Computer modern fonts. Скажем несколько необходимых для дальнейшего слов о том, в каком виде хранятся **TeX'**овские шрифты. Как мы неоднократно отмечали выше, в процессе верстки текста **TeX'**у неважно, как реально выглядят символы. Все, что в этот момент используется — информация о том, сколько места надо оставить на каждый символ (плюс еще некоторые тонкости: например, какие символы составляют лигатуру, подобно тому как сочетание ?' дает на печати ?, между какими символами предусмотрен кернинг, и т. п.). Все эти данные содержатся в файлах с расширением **tfm**. Реальное начертание символов становится важно при просмотре, печати, и вообще в тех случаях, когда в игру вступают **dvi**-драйверы. Поэтому информация о начертании символов хранится в отдельных файлах. Файлы, в которых записана форма символов, мы будем называть **pk**-файлами, так как они обычно (хотя и не всегда) имеют расширение **pk**.

Одному **tfm**-файлу может соответствовать много разных **pk**-файлов. Дело в том, что в шрифтах, созданных с помощью **METAFONT**'а, предусмотрена возможность их масштабирования. В процессе трансляции исходного текста **TeX** при работе с масштабированным (скажем, увеличенным в два раза) шрифтом просто умножает на заданный коэффициент размеры символов, взятые из уже существующего **tfm**-файла. А вот при печати или просмотре понадобится уже и новый **pk**-файл, соответствующий масштабированному шрифту.

ЛАТ $\mathrm{\TeX}$ предусматривает возможность включения в ваш документ шрифтов, не входящих в стандартный комплект. Для этого используется команда **\newfont**. Ее формат таков:

`\newfont{команда}{описание_шрифта}`

Здесь *команда* — это команда для переключения на добавляемый вами шрифт. Придуманное вами имя этой команды должно подчиняться обычным ТЕХ'овским правилам и не должно совпадать с именами уже существующих команд. Что же до *описания_шрифта*, то в простейшем виде это — просто имя *tfm*-файла, соответствующего данному шрифту. Вот пример. В свое время Дональд Кнут шутки ради разработал такой причудливый шрифт, называемый *cmff10*. Чтобы пользоваться этим шрифтом в своем тексте, включите в преамбулу строку

`\newfont{\косой}{cmff10}`

и вы сможете писать тексты вроде

Буквы выглядят слегка кособокими.	Буквы выглядят слегка {\косой кособокими.}
-----------------------------------	---

По сути дела, `\newfont` определяет новую команду для переключения шрифтов, похожую на привычные вам команды вроде `\Large` (в частности, действие этой новой команды заканчивается по выходе из той группы, внутри которой она была дана). Есть, однако, два существенных различия. Во-первых, в отличие от команд наподобие `\Large`, такие команды не меняют ни интервалов между строками, ни размеров невидимой линейки, создаваемой командой `\strut`. Во-вторых, внутри математической формулы такая команда вообще никакого действия не оказывает.

Выше говорилось, что в ТЕХ'е можно использовать масштабированные шрифты. Чтобы подключить такой шрифт с помощью команды `\newfont`, надо задать требуемое увеличение или уменьшение во втором аргументе команды `\newfont`. Оно задается с помощью ТЕХ'овского «ключевого слова» `scaled` (без backslash!), за которым следует коэффициент масштабирования, умноженный на 1000 (после умножения коэффициента на 1000 должно получиться целое число). Например, для подключения шрифта, увеличенного в два с половиной раза, надо после имени *tfm*-файла написать `scaled 2500`, а для шрифта, размеры которого уменьшены на 30%, надо написать `scaled 700`. Будем надеяться, что у вас есть либо *pk*-файлы, соответствующие масштабированным таким образом шрифтам, либо программа *METAFONT*, с помощью которой эти файлы можно сгенерировать (при условии, что вы располагаете еще и «исходными текстами» шрифтов).

В реально используемых в ТЕХ'е шрифтах коэффициенты 2.5 или 0.7 на самом деле не употребляются. Применяемые на практике коэффициенты увеличения образуют геометрическую прогрессию со знаменате-

Таблица А.1. Прямой светлый шрифт (cmr10 и др.)

	"00	"01	"02	"03	"04	"05	"06	"07	"08	"09	"0A	"0B	"0C	"0D	"0E	"0F
"00	Г	Δ	Θ	Λ	Ξ	Π	Σ	Τ	Φ	Ψ	Ω	ff	fi	fl	ffi	fff
"10	ı	J	`	'	-	-	-	°	,	ß	æ	œ	ø	Æ	Œ	Ø
"20	'	!	"	#	\$	%	&	,	()	*	+	,	-	.	/
"30	0	1	2	3	4	5	6	7	8	9	:	;	i	=	¿	?
"40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
"50	P	Q	R	S	T	U	V	W	X	Y	Z	[“	”	^	‘
"60	'	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
"70	p	q	г	s	t	u	v	w	x	y	z	-	-	"	~	"

лем 1.2. Для таких увеличений можно использовать более удобные обозначения, при помощи команды `\magstep`: вместо `scaled 1200` можно написать `scaled \magstep 1`, вместо `scaled 1440` — `scaled \magstep 2` (поскольку $1.2^2 = 1.44$), и т. д. (максимально возможное значение — `\magstep 5`). Можно также сказать `\magstephalf`, что задает увеличение в $\sqrt{1.2}$ раза.

Можно также задавать увеличение не в явном виде, а сообщить *TeX*'у требуемый «характерный размер» шрифта. Для этого надо во втором аргументе команды `\newfont` написать после имени *tfm*-файла масштабируемого шрифта

at *размер*

где *размер* — требуемый «характерный размер», заданный обычным образом в *TeX*'овских единицах длины или через *TeX*'овские параметры длины. Если основной шрифт вашего документа имеет кегль 10, то характерный размер разумно выбирать равным *10pt*, если 11 или 12, то *11pt* или *12pt* соответственно. Например, существует (не входящий в *LATeX*'овский стандартный комплект) шрифт *wasy10*, содержащий различные причудливые символы. Если вы набираете текст 12-го кегля, то для подключения шрифта *wasy10* в соответствующем размере лучше написать в преамбуле

```
\newfont{\wasy}{wasy10 at 12pt}
```

Таблицы кодов символов в основных текстовых шрифтах, используемых *LaTeX*'ом, представлены в табл. А.1 и А.2. Необходимость знать эти коды возникает сравнительно редко, например, когда вы пользуетесь *LaTeX*'овской командой `\symbol` (см. с. 88). В таблицах не представлены коды русских букв, поскольку они зависят от используемой

Таблица А.2. Шрифт typewriter (cmtt10 и др.)

"00	Г	Д	Ђ	Љ	Ң	ң	Ҥ	Ҧ	Ҩ	ҩ	↑	↓	'	і	ѣ
"10	ј	ј	‘	’	—	—	—	—	,	Ѡ	ѡ	ø	Ѐ	Ӯ	Ը
"20	ւ	!	"	#	\$	%	&)	(*	+	,	-	.	/
"30	Օ	1	2	3	4	5	6	7	8	9	:	;	<	=	> ?
"40	ܰ	ܱ	ܲ	ܳ	ܴ	ܵ	ܶ	ܷ	ܸ	ܹ	ܻ	ܼ	ܻ	ܼ	ܽ
"50	ܫ	ܬ	ܭ	ܮ	ܯ	ܰ	ܱ	ܲ	ܳ	ܴ	ܵ	ܶ	ܷ	ܸ	ܹ
"60	ܪ	ܫ	ܬ	ܭ	ܮ	ܰ	ܱ	ܲ	ܳ	ܴ	ܵ	ܶ	ܷ	ܸ	ܹ
"70	ܫ	ܬ	ܭ	ܮ	ܰ	ܱ	ܲ	ܳ	ܴ	ܵ	ܶ	ܷ	ܸ	ܹ	ܹ

русификации. Коды символов даны в шестнадцатеричной записи: первая цифра записи определяет строку, в которой стоит символ, вторая — столбец. Например, в прямом светлом шрифте (табл. А.1) буква ё стоит на пересечении строки "10 и столбца "0A ; значит, ее код равен 1A в шестнадцатеричной записи и $1 \cdot 16 + 10 = 26$ в десятичной записи, и эту букву можно напечатать командами `\symbol{"1A}` или `\symbol{26}`. Прямой светлый шрифт из табл. А.1 является текущим в начале работы L^AT_EX'a. Соответствующие tfm-файлы называются cmr10, cmr11 и т. п.: последние две цифры совпадают с «характерным размером» шрифта, выраженным в пунктах.

Расположение символов в наклонном и жирном шрифтах точно такое же, как в прямом светлом; в курсивном шрифте (названия tfm-файлов cmti10 и др.) на месте знака \$ стоит знак £.

В шрифте typewriter (вызывается командой `\tt`; tfm-файлы называются cmtt10 и др.) раскладка символов иная, чем в остальных текстовых шрифтах, так что для него мы приводим отдельную таблицу.

Приложение Б

ЛАТЕХ и другие макропакеты

В предисловии мы уже отмечали, что наряду с \LaTeX 'ом употребительны и такие макропакеты на базе языка \TeX , как Plain \TeX и \AMS-\TeX . Друг с другом эти макропакеты несовместимы: обработать \LaTeX 'овский файл с помощью Plain \TeX 'а не удастся. Тем не менее, во всех трех макропакетах немало общего; в частности, если вы освоили \LaTeX , то сможете готовить несложные файлы и в Plain \TeX 'е. Мы не будем пытаться создать сколько-нибудь полное представление о возможностях Plain \TeX 'а или \AMS-\TeX 'а, но только сообщим, какие \LaTeX 'овские команды можно использовать в Plain \TeX 'е (и отчасти в \AMS-\TeX 'е).

Основной принцип \TeX 'а полностью применим и к Plain \TeX 'у, и к \AMS-\TeX 'у: слова разделяются пробелами (неважно, в каком количестве), абзацы — пустыми строками (неважно, в каком количестве), внутритеческие формулы выделяются знаками доллара, а выключные — парами знаков доллара.

Спецсимволы в Plain \TeX 'е те же самые, что и в \LaTeX 'е (с. 16); отличие в том, что команды $\{$ и $\}$ в Plain \TeX 'е можно применять только в формулах. В \AMS-\TeX 'е к числу спецсимволов относится и \bullet ; для его печати применяется команда \textbullet .

Элементарные правила набора формул, перечисленные в гл. I, применимы в Plain \TeX 'е и \AMS-\TeX 'е, со следующими двумя исключениями:

В \LaTeX 'е пишем: А в Plain \TeX 'е надо:

$\text{\frac}{12}{34}$ $\{12\over 34\}$
 $\text{\sqrt}{10}$ $\text{\root 10 \of}{1024}$

В примере с командой \over обратите внимание, что числителем дроби

будет весь текст от открывающей фигурной скобки до `\over`, а знаменателем — весь текст от `\over` до закрывающей скобки — ср. обсуждение команды `\choose` на с. 66. В *AMS-TeX*'е можно пользоваться и знакомой из *L^AT_EX*'а командой `\frac`). Аналогично, показателем корня будет весь текст от `\root` до `\of`. Для обычного квадратного корня (показатель которого, как известно, не пишется) в Plain *TeX*'е применяется знакомая нам команда `\sqrt`.

Все математические спецзнаки, перечисленные в таблицах разд. II.1, можно применять и в Plain *TeX*'е и в *AMS-TeX*'е. В *AMS-TeX*'е можно также пользоваться командами `\Bbb` (эквивалент `\mathbb` из *L^AT_EX*'а 2_ε) и `\frak` (эквивалент `\mathfrak` из *L^AT_EX*'а 2_ε), а также командами, перечисленными в таблицах приложения Д. Более того, при наборе формул в Plain *TeX*'е и *AMS-TeX*'е можно пользоваться почти всеми средствами, описанными в гл. II, со следующими исключениями:

- не определены команды `\:, \pounds, \stackrel` и `\lefteqn`;
- при включении текста в формулы следует пользоваться командой `\hbox` вместо `\mbox`;
- нумеровать формулы надо вручную, с помощью `\eqno` или `\leqno`;
- команды смены шрифтов в формуле, специфичные для *L^AT_EX*'а 2_ε, в Plain *TeX*'е применять нельзя; что же до *AMS-TeX*'а, то в нем, напротив, нельзя применять в формулах знакомые по *L^AT_EX*'у команды `\rm, \bf` и т. д.: используйте команды `\roman` и `\bold`, работающие так же, как `\mathrm` и `\mathbf` из *L^AT_EX*'а 2_ε;
- нельзя пользоваться *TeX*'овским окружением `array`.

Последнее из перечисленных ограничений наиболее серьезно. По поводу способов набора матриц в *AMS-TeX*'е отсылаем читателя к книге [6]; что же до Plain *TeX*'а, то вот два примера, руководствуясь которыми читатель, видимо, сможет справиться с этой задачей хотя бы в простых случаях:

```
 $$\matrix{ a_{11} & a_{12} & \dots & a_{1n} & a_{\{11}\&a_{\{12}\&\ldots&a_{\{1n}\}\cr a_{21} & a_{22} & \dots & a_{2n} & a_{\{21}\&a_{\{22}\&\ldots&a_{\{2n}\}\cr \vdots & \vdots & \ddots & \vdots & \vdots\&\vdots\&\ddots\&\vdots\cr a_{n1} & a_{n2} & \dots & a_{nn} & a_{\{n1}\&a_{\{n2}\&\ldots&a_{\{nn}\}\cr } $$
```

Здесь `\cr` — *TeX*'овская команда, играющая в Plain *TeX*'е ту же роль, что `\backslash` в *TeX*'овских окружениях `array` и `tabular`.

А вот еще одна полезная конструкция из Plain *TeX*'а:

$$|x| = \begin{cases} x, & x \text{ положительно;} \\ 0, & x = 0; \\ -x, & x \text{ отрицательно.} \end{cases}$$

\$\$ |x|=\cases{ x, & x\text{ положительно;}\cr 0, & x=0;\cr -x,&x\text{ отрицательно.}\cr} \$\$

Обратите внимание, что текст до `&` обрабатывается в математическом режиме, а текст между `&` и `\cr` — в текстовом.

Теперь обсудим, с чем может столкнуться любитель *LATEX*'а вне математических формул. Большая часть средств, описанных в первых трех разделах гл. III, применима в Plain TeX'е и *AMS-Tex*'е. Есть, однако, несколько неприятных мелочей. По поводу *AMS-Tex*'а мы опять отсылаем читателя к книге [6], что же до Plain TeX'а, то вот список тех мест, где рецепты из первых трех разделов гл. III не годятся:

- команды `\,, \&, \symbol` и `\fbox` не определены; `\underline` можно пользоваться только в формулах, но не в тексте;
- командой `\,` можно пользоваться только в формулах;
- для печати многоточия в тексте можно использовать только команду `\dots`.

Если углубиться в ту же главу дальше, то запретов станет больше:

- команда `\hspace` не определена;
- команды для изменения размера шрифта не определены;
- команда `\em`, а также команды *LATEX*'а `2e` для смены шрифта не определены;
- *LATEX*'овские команды для сносок в Plain TeX'е непригодны;
- для предотвращения переносов в слове пользуйтесь не `\mbox`, а конструкцией с `\-` в конце (с. 109);
- *LATEX*'овские команды для принудительного разрыва строки или для запрета такого разрыва в Plain TeX'е не работают;
- из всех средств, перечисленных в разд. III.8, в Plain TeX'е можно пользоваться только следующими командами: `\par`, `\noindent`, `\smallskip`, `\medskip`, `\bigskip`;
- *LATEX*'овской командой `\rule` пользоваться нельзя.

Всю остальную часть нашей книги можно при работе с Plain *TeX*'ом благополучно забыть.

Для пользующихся Plain *TeX*'ом отметим, что командой `\proclaim` можно пользоваться и в *LATEX*'е, хотя мы и не описывали ее в этой книжке. Это может облегчить перевод текстов из Plain *TeX*'а в *LATEX* (если вы собираетесь и дальше работать над получающимся *LATEX*'овским файлом, то лучше все же заменить `\proclaim` на окружение типа «теорема», чтобы иметь возможность пользоваться автоматической нумерацией).

В Plain *TeX*'е не предусмотрено специальной команды, начинающей файл: можно сразу начинать писать текст; завершаться файл для Plain *TeX*'а должен командой `\bye`. Файл для *AMS-TeX*'а может выглядеть так:

```
\documentstyle{amsppr}
\document
...
\enddocument
```

Будем надеяться, что читатель этой книги сможет теперь набирать простые тексты в Plain *TeX*'е. Что же касается *AMS-TeX*'а, то читатель может обратиться к книге [6], в которой все изложено в весьма доступной форме. Однако лучше всего перейти на *LATEX* 2 ϵ , после чего работать с *AMS-LATEX*'овскими стилевыми пакетами или классами документов: при этом в ваших руках будут все возможности *AMS-TeX*'а вкупе со всеми удобствами *LATEX*'а. См. по этому поводу приложение Е.

Приложение В

Импорт графики в пакете *emTeX*

Как уже отмечалось в гл. V, включение графических файлов в текст задается командой `\special`, синтаксис которой не стандартизирован, а зависит от используемой реализации *TeX*'а (точнее, от используемых *dvi*-драйверов). К счастью или к несчастью, на сегодняшний день стандартом в нашей стране являются компьютеры типа IBM PC. Наиболее популярная (и наиболее удачная) реализация *TeX*'а на этой платформе — свободно распространяемый пакет *emTeX*, написанный Эберхардом Маттесом (Eberhard Mattes) из Германии. В настоящем приложении мы расскажем, как работать с графикой, пользуясь командой `\special` и *dvi*-драйверами из этого пакета.

Все *emTeX*'овские команды `\special` удобно помещать на псевдорисунках (окружение `picture`) с помощью команды `\put`. Ничто не мешает в том же окружении `picture` использовать, наряду со `\special`, и команды `\put` с уже знакомыми по гл. V аргументами, и вообще любые команды, допустимые в псевдорисунках. Например, это бывает необходимо, чтобы нанести на рисунок надписи.

Основная операция, связанная с графикой, — включение в документ изображения, заданного в виде графического файла. Все *emTeX*'овские *dvi*-драйверы понимают графические файлы типа `psx`, `bmp` и `msp`. Чтобы включить, например, в псевдорисунок изображение кошки, записанное в графическом файле `catsy.bmp`, надо включить в это окружение такую команду:

```
\put(x,y){\special{em:graph catsy.bmp}}
```

Здесь (x,y) — координаты левого верхнего угла рисунка. Аргумент команды `\special` должен начинаться с `em:graph`, после пробела следует имя включаемого графического файла (вместе с расширением).

В общем случае *TeX* считает, что команда `\special` генерирует блок нулевой ширины, высоты и глубины; *emTeX*'овские *dvi*-драйверы считают, что точка отсчета этого блока находится в левом верхнем углу рисунка.

Наряду с импортом графики, *emTeX*'овские *dvi*-драйверы умеют рисовать отрезки произвольного наклона (в *LATEX*'овской псевдографике, как вы помните, репертуар возможных наклонов скучен). Чтобы нарисовать отрезок толщиной в 1 пункт, соединяющий на псевдорисунке точки с координатами (2, 3) и (11, 5), можно действовать так:

```
\put(2,3){\special{em:point 1}}
\put(11,5){\special{em:point 2}}
\special{em:line 1,2,1pt}\%
```

Общие правила таковы. Команда

```
\special{em:point n}
```

на печати сама по себе ничего не дает, но определяет точку с номером *n*, где *n* — целое число от 1 до 32767 (на разных страницах номера точек могут повторяться). Расположение точки определяется обычной *LATEX*'овской командой `\put`. Команда

```
\special{em:line n, m, w}
```

выводит на печать отрезок толщиной *w*, соединяющий точки с номерами *m* и *n*. Толщину можно задавать в любых *TeX*'овских единицах (см. с. 25). Параметры со значением длины, типа `\linethickness`, использовать в этом месте нельзя.

Причина в том, что во время трансляции аргумент команды `\special` просто переносится в *dvi*-файл (дословно), а интерпретируется он уже *dvi*-драйверами, которые имеют дело только с *dvi*-файлом и знать ничего не знают ни о *LATEX*'овских параметрах, ни даже о том, каким *TeX*'овским макропакетом создавался *dvi*-файл.

Обратите внимание, что `\special{em:line...}` ставится не в аргументе команды `\put`, а сама по себе; в нашем примере мы закончили соответствующую строку знаком процента, чтобы удалить пробел, генерируемый концом строки (после команды `\put` такие предосторожности не требуются).

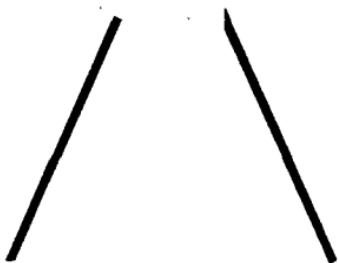
Можно после `em:line` указать только номера точек, опустив указание на толщину отрезка. Тогда по умолчанию отрезок будет иметь толщину 0.4 pt.

Можно изменить «толщину отрезков по умолчанию», дав команду

```
\special{em:linewidth толщина}
```

Здесь *толщина* опять может быть задана в любых TeX'овских единицах.

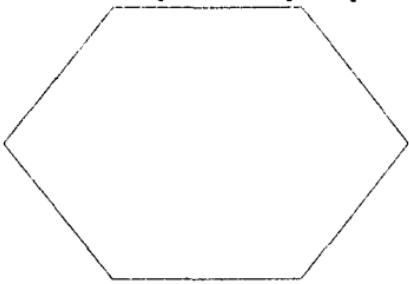
Отрезки (по крайней мере достаточно толстые) выглядят по-разному в зависимости от того, как обрезаны их края. В аргументе команды `\special` можно передать соответствующую информацию для dvi-драйвера. Именно, для этого надо после одного из (или обоих) номеров точек поставить букву `h`, `v` или `p`. Буква `h` означает, что соответствующий конец отрезка будет обрезан горизонтально, `v` — что вертикально, `p` — что перпендикулярно проведенному отрезку (так и будет по умолчанию, если никаких букв после номеров точек не добавлять). Вот пример на использование этих средств:



```
\begin{picture}(120,100)
\put(0,0){\special{em:point 10}}
\put(40,90){\special{em:point 20}}
\put(80,90){\special{em:point 30}}
\put(120,0){\special{em:point 40}}
\special{em:line 10h,20p,3pt}%
\special{em:line 30v,40,3pt}%
\end{picture}
```

Обратите внимание, что буквы `h`, `v` или `p` после номеров точек ставятся при указании этих номеров вместе с `em:line`, но не `em:point`.

Можно также воспользоваться другим приемом рисования отрезков, особенно удобным, если надо изобразить ломаную линию. Для этого применяются команды `\special` с аргументами `em:moveto` и `em:lineto`. Команду `\special` с каждым из этих аргументов можно разместить на псевдорисунке обычной командой `\put`. Работу этих `\special`'ов удобно посмотреть на примере:



```
\begin{picture}(150,100)
\put(0,50){\special{em:moveto}}
\put(40,100){\special{em:lineto}}
\put(110,100){\special{em:lineto}}
\put(150,50){\special{em:lineto}}
\put(110,0){\special{em:lineto}}
\put(40,0){\special{em:lineto}}
\put(0,50){\special{em:lineto}}
\end{picture}
```

Чтобы понять, как устроен этот пример, удобно представлять себе, что dvi-драйвер рисует ломаную линию пером. При этом по команде `\special{em:moveto}` он ничего не рисует, но заносит перо над

бумагой (в той точке, куда мы поместили `\special{em:moveto}` с помощью команды `\put`); по команде `\special{em:lineto}` он опускает занесенное перо на бумагу и ведет его до точки, в которую мы поместили `\special{em:lineto}` с помощью команды `\put`. Если дальше будет еще одно `\special{em:lineto}`, то *dvi*-драйвер проведет еще один отрезок (начиная с того места, где он остановился в предыдущий раз), а если дальше будет `\special{em:moveto}`, то *dvi*-драйвер просто перенесет перо в новое место, ничего не рисуя.

Подчеркнем еще раз, что все описанные средства применимы только в том случае, если вы пользуетесь *dvi*-драйверами из пакета *emTeX*. Если вы обработаете *LATeX*'овский файл, в котором используются команды `\special` с аргументами, описанными выше, с помощью другой реализации *TeX*'а, то при трансляции текста никаких проблем не будет, а вот при печати или просмотре ни включенных графических файлов, ни *emTeX*'овских отрезков и ломаных вы не увидите.

Приложение Г

Программа makeindex

Как мы отмечали в гл. IV, L^AT_EX'овские команды \index и \makeindex создают idx-файл, являющийся полуфабрикатом для предметного указателя. В этом приложении мы расскажем о программе, перерабатывающей этот полуфабрикат в полноценный предметный указатель, который можно уже включить в ваш L^AT_EX'овский файл. Эта свободно распространяемая программа, называемая makeindex, входит в комплект поставки пакета emT_EX (см. приложение B); свободно распространяется и ее исходные тексты на языке C (тем самым эта программа доступна и для работающих под операционной системой UNIX). Специально на русский язык программа makeindex рассчитана не была, однако русские тексты в наиболее распространенной на IBM PC «альтернативной» кодировке она обрабатывает более или менее правильно. Оговорка связана с тем, что неверно устанавливается соответствие прописных и строчных русских букв; вариант программы, полностью приспособленный к русским текстам в альтернативной кодировке, можно получить, например, в ассоциации СугTUG (см. рекламу в конце книги) или у А. Шеня (shen@ium.ips.ras.ru).

1. Простейшие средства

Версия программы makeindex для IBM PC может называться, в зависимости от типа процессора, makeindx или mklidx32. Версия для системы UNIX называется просто makeindex. В дальнейшем мы будем считать, что программа называется makeindx.

В простейшем случае программа makeindex вызывается так:

makeindx -c исходный_файл

Если *исходный_файл* имеет расширение *idx*, то это расширение можно не указывать. Исходным файлом для программы makeindex является

idx-файл, создаваемый L^AT_EX'ом (см. разд. IV.7.4). В результате работы программы makeindex появится файл с тем же именем, что у исходного файла, и расширением ind. Это — готовый файл для предметного указателя, который остается только включить в ваш документ с помощью команды \input. Создается также файл с тем же именем и расширением ilg. Это — протокол работы программы makeindex.

При обработке idx-файла программой makeindex пробелы в начале и конце записей будут игнорироваться, а два и более пробелов будут рассматриваться как один. Благодаря этому записи \index{Кошка} и \index{ Кошка} будут рассматриваться как относящиеся к одному и тому же ключевому слову. Если бы такого игнорирования пробелов не было, то программа makeindex отвела бы в указателе отдельную строку для кошки, начинающейся с пробела. Чтобы отменить игнорирование лишних пробелов, надо запустить программу makeindex без ключа -c.

Если не предпринимать специальных мер, то все записи в ind-файле, созданном программой makeindex, будут равноправны — все они будут вводиться командой \item. Чтобы предметный указатель был устроен иерархически, как в примере на с. 161, надо в аргументе команды \index после заглавного слова поставить восклицательный знак, а после него — подчиненное ему слово. Возможно также подчинение второго порядка — тогда нужен еще один восклицательный знак. Вот пример:

```
Многие люди любят домашних кошек.\index{Кошки!домашние}
...
Ваша киска\index{Кошки!домашние!уход} купила бы...
...
Хорошо также иметь собаку.\index{Собаки}
...
Мало кто рискнет держать дома такую дикую
кошку,\index{Кошки} как тигр. Пудель\index{Собаки}
гораздо безопаснее.
```

При обработке этого файла L^AT_EX'ом получится idx-файл³²; в результате обработки idx-файла программой makeindex получится ind-файл, включающий в себя, в частности, следующее (предположим, что наши команды \index попали на страницы с номерами 2, 7, 8, а две последние — на страницу 9):

```
\begin{theindex}
```

```
...
```

³²В том, конечно, случае, если в преамбуле была команда \makeindex.

```

\item Кошки, 9
  \subitem домашние, 2
    \subsubitem уход, 7
...
  \item Собаки, 8, 9
...
\end{theindex}

```

Из сказанного следует, что, если вы намереваетесь обрабатывать *idx*-файл программой *makeindex*, восклицательный знак в аргументе команд *\index* будет иметь особый статус. Чтобы программа *\makeindex* восприняла восклицательный знак просто как типографский значок, надо в аргументе *\index* предварить его знаком кавычки ":

```

\index{Восклицательный знак ("!)}
\index{Междометия!Эх"!}

```

Эти аргументы команд *\index* дословно скопируются в *idx*-файл, а после его обработки программой *makeindex* в *ind*-файл запишется примерно вот что:

```

\item Восклицательный знак (!), 14
\item Междометия
  \subitem Эх!, 6

```

Наряду с восклицательным знаком, особый статус с точки зрения программы *makeindex* имеет символ © («коммерческое at»), вертикальная черточка | (в следующем разделе вы узнаете, в чем этот статус заключается), а также сама кавычка ". Если вы хотите употребить один из этих четырех значков в аргументе команды *\index* просто как символ, не вкладывая в него специального смысла, надо поставить перед ним кавычку ".

Исключение: кавычку, входящую в состав *TeX*'овской команды \"', можно (и нужно) записывать без предосторожностей:

```

\index{Кавычка ("")}
\index{\\"Елочка}

```

При этом соответствующие записи в *idx*-файле могут получиться такими:

```

\item \"Елочка, 8
\item Кавычка ("'), 14

```

2. Тонкости

Для каждого ключевого слова программы makeindex собирает все относящиеся к нему номера страниц и записывает их в *ind*-файле после этого слова через запятую. Если при этом попадутся три или более идущих подряд номера страниц, то в *ind*-файл будут записаны только первый и последний из этих номеров, через короткое тире (en-dash). Такие пары страниц через короткое тире можно организовывать и вручную. Пусть, например, в какой-то части вашего текста все время идет речь о кошках. Тогда можно в начале этой части написать

```
\index{Кошки|()}
```

а в конце —

```
\index{Кошки|})
```

Если первая из этих команд попала на страницу 9, а вторая — на страницу 77, то, после обработки *idx*-файла программой makeindex, в *ind*-файл попадет запись

```
\item Кошки, 9--77
```

Команды `\index{Кошки}`, попавшие между страницами 9 и 77, будут при этом проигнорированы.

При сортировке программа makeindex принимает во внимание не только буквы, но и спецзнаки, записанные в аргументе команды `\index`. Иногда это нежелательно: если в тексте имеются команды `\index{Лист}` и `\index{{\bf Ящерица}}`, то ящерица окажется в *ind*-файле раньше аиста, так как makeindex будет считать, что запись для нее начинается с символа {, который идет раньше всех русских букв. Чтобы избежать такого рода неприятностей, предусмотрена возможность по отдельности задать слово, которое будет участвовать в сортировке, и текст, который будет реально записан в *ind*-файл. В приведенном выше примере следовало бы написать

```
\index{Ящерица@\bf Ящерица})
```

Теперь ящерица попадет туда же, куда и все прочие слова на букву я, но при этом будет напечатана жирным шрифтом. Общее правило такое: если в аргументе команды `\index` присутствует символ @, то при сортировке учитывается только то, что написано левее него, а в *ind*-файл записывается только то, что правее него. Можно задавать отдельные тексты для сортировки и для печати не только для основного заглавного слова, но и для слов, ему подчиненных:

```
\index{Ящерицы@\bf Ящерицы}!Игуана@\bf Игуана}
```

Напоминание: если перед @ или ! стоит кавычка, то эти значки рассматриваются просто как символы.

Следующая возможность программы makeindex — создать особое оформление для номеров отдельных страниц. Например, в предметном указателе к книге TeXbook [3] номера страниц, на которых содержится наиболее подробное описание заглавного термина, подчеркнуты. Вот как это можно сделать с помощью программы makeindex. Если одно из мест в тексте, где говорится о кошках, вы считаете особо важным, то можно поставить в этом месте команду

```
\index{Кошки}\underline{}
```

Предположим, что эта команда попала на страницу 100, и, кроме того, в тексте были две команды \index{Кошки}, попавшие на страницы 15 и 47. Тогда, после обработки idx-файла программой makeindex, в ind-файле появится строка:

```
\item Кошки, 15, 47, \underline{100}
```

Общее правило таково: команда \index{XXX\abcd} порождает в ind-файле строку

```
\item XXX, \abcd{y}
```

(здесь *y* — номер страницы). Уточнения: среди символов *abcd* не должно быть круглых скобок (сочетания | (и)) имеют, как было сказано выше, иной смысл); наряду с \abcd{y} могут присутствовать и другие номера страниц.

Если вы хотите номер страницы не подчеркнуть, а набрать, скажем, жирным шрифтом, то придется записать в аргументе \index после | имя команды с одним аргументом, печатающей свой аргумент жирным шрифтом. В L^AT_EX'е 2_ε такая команда есть в готовом виде (если вы не забыли, она называется \textbf{}); при пользовании L^AT_EX'ом 2.09 придется создать ее самостоятельно, написав в преамбуле определение наподобие

```
\newcommand{\TBF}[1]{\bf #1}
```

3. Настройка программы makeindex

В предыдущих разделах мы объясняли, как и какую информацию можно передавать программе makeindex. Теперь объясним, как добиться того, чтобы она обрабатывала эту информацию по-иному.

Во-первых, можно задать по своему усмотрению имя файла, в который makeindex запишет отсортированный и обработанный idx-файл. Для этого надо воспользоваться ключом **-o**:

```
makeindx -c -o выходной_файл исходный_файл
```

Чтобы задать отличное от стандартного имя файла с протоколом трансляции, надо аналогичным образом воспользоваться ключом **-t**.

Наконец, самое существенное: можно запустить программу makeindex вместе со *стилевым файлом*, в котором можно дать программе указания по поводу вида, в котором будет записан отсортированный и обработанный idx-файл. Написав подходящий стилевой файл для makeindex, можно радикально изменить вид ind-файла, так, что он вообще не будет иметь ничего общего с *Л^AT_EX'*овским файлом (это может иметь смысл: makeindex задуман как программа широкого профиля, пригодная не только для *Л^AT_EX'*а). Как добиться столь революционных изменений, мы обсуждать не будем (интересующиеся могут узнать все подробности из оригинальной документации), но о некоторых вещах, полезных именно для *Л^AT_EX'*а, расскажем.

Чтобы подключить стилевой файл к makeindex, надо запустить эту программу с ключом **-s**, после которого, через пробел, указывается имя стилевого файла (по традиции он имеет расширение *ist*). Если стилевой файл называется *mystyle.ist*, то можно сказать так:

```
makeindx -c -s mystyle.ist исходный_файл
```

Теперь обсудим, что можно менять с помощью стилевого файла. Как мог заметить читатель, программа makeindex автоматически записывает строку

```
\begin{theindex}
```

в начало ind-файла и

```
\end{theindex}
```

в его конец. Часто требуется, чтобы в начало или конец ind-файла автоматически записывалось что-то еще (команда *\sloppy* в начало, например). Для того, чтобы после *\begin{theindex}* было на отдельной строке написано еще и *\sloppy*, надо в стилевом файле написать так:

```
preamble "\begin{theindex}\n\sloppy\n"
```

Здесь `preamble` — имя стилевого параметра, определяющего, что записывается в начало всякого `ind`-файла. Остальной текст — содержание этой записи. Правила записи в индексном файле несколько напоминают правила записи строковых констант из языком С:

- строковая константа, задающая стилевой параметр, ограничена с обеих сторон знаком " (кавычка);
- строковая константа, задающая стилевой параметр, может реально состоять и из нескольких строк; место, где кончается одна строка и начинается другая, обозначается `\n` (конец строки воспринимается просто как пробел и не означает конца строки в `ind`-файле);
- если в строковую константу должны входить символы `\` или `"`, то их надо обозначать `\\"` и `\"` соответственно, а все остальные символы набираются непосредственно.

Параметр `postamble` определяет, что записывается в конце `ind`-файла. По умолчанию это

```
"\n\n\\end{theindex}\n"
```

(иными словами: начать с новой строки, одну строку пропустить, написать `\end{theindex}`, строку закончить).

Следующие три параметра определяют, чем отделяются номера страниц от ключевых слов: `delim_0` — для ключевых слов «верхнего уровня», `delim_1` и `delim_2` — для слов первого и второго уровня подчинения. По умолчанию все три этих параметра определены как `,`, `"` (запятая и пробел), вследствие чего номера страниц отделяются от слов запятыми. В русских текстах эти запятые ставить не принято, поэтому все три этих параметра стоит переопределить на `" "`:

```
delim_0 " "
delim_1 " "
delim_2 " "
```

Наконец, параметр `group_skip` определяет, что записывается в `ind`-файл между группами слов, начинающихся на одну букву. Значение по умолчанию — `"\n\n \\indexspace\n"` («пропустить строку, написать `\indexspace` и начать с новой строки»).

Приложение Д

Шрифты Американского математического общества

Математических знаков в стандартном L^AT_EX'овском наборе очень много, но порой и их не хватает. Если вы пользуетесь L^AT_EX'ом 2_E, то есть возможность воспользоваться дополнительными математическими знаками, разработанными Американским математическим обществом (сокращенно AMS — American Mathematical Society). Для этого надо подключить стилевой пакет *amssymb*. При этом вы сможете пользоваться в формулах шрифтами, определенными в пакете *amsfonts* (см. разд. II.2.4), но наряду с этим вам станут доступны больше полутора сотен новых математических символов. Чтобы не запутаться в этом обилии, мы разобьем их группы, как в гл. II. Впрочем, это разбиение делается не только для удобства восприятия: как мы уже отмечали в разд. II.4.4, расстановка интервалов в формулах зависит от того, к какой группе (бинарная операция, бинарное отношение, обыкновенный символ и т. д.) относится математический символ.

Начнем с бинарных операций. Стилевой пакет *amssymb* дает возможность воспользоваться такими символами, относящимися к этой категории:

⊕ \boxplus	⊖ \boxminus	⊗ \boxtimes
□ \boxdot	· \centerdot	⊓ \veebar
⊜ \barwedge	⊜ \doublebarwedge	⊟ \Cup
⊠ \Cap	⊢ \curlywedge	⊣ \curlyvee
⊤ \leftthreetimes	⊥ \rightthreetimes	⊦ \dotplus
⊲ \intercal	⊸ \circledcirc	⊹ \circledast
⊖ \circleddash	⊸ \lessdot	⊹ \gtrdot
⊸ \ltimes	⊸ \rtimes	⊸ \smallsetminus
* \divideontimes		

Следующим по очереди идет огромное количество бинарных отношений:

\sqsubset	\sqsupset
\vartriangleleft	\vartriangleleft
\triangleleft	\triangleleft
\trianglelefteq	\trianglelefteq
\vartriangleright	\vartriangleleft
\triangleright	\triangleleft
\trianglerighteq	\trianglelefteq
\Vdash	\Vdash
\triangleq	\Vdash
\lessim	\precsim
\eqslantless	\lessapprox
\curlyeqprec	\eqslantgr
\preccurlyeq	\curlyeqsucc
\leqslant	\leqq
\backprime	\lessgtr
\fallingdotseq	\risingdotseq
\geqq	\succcurlyeq
\gtrless	\geqlant
\blacktriangleright	\between
\vartriangle	\blacktriangleleft
\lesseqtr	\eqcirc
\lesseqgtr	\gtreqless
\varpropto	\gtreqqless
\smallfrown	\smallsmile
\Supset	\Subset
\supseteqq	\subsetneqq
\Bumpeq	\bumpeq
\ggg	\lll
\backsimeq	\pitchfork
\lvertneqq	\backsimeq
\lneqq	\gvertneqq
\lneq	\gneqq
\precsim	\gneq
\lnsim	\succnsim
\precneqq	\gnsim
\precnapprox	\succcneqq
\lnapprox	\succcnapprox
\diagup	\gnapprox
\varsubsetneq	\diagdown
\subsetneq	\varsupsetneq
\subsetneqq	\supsetneq
\varsubsetneqq	\varsupsetneqq
\subsetneqq	\varsupsetneqq

\approx	<code>\eqsim</code>	\shortmid	<code>\shortmid</code>
\shortparallel	<code>\shortparallel</code>	\thicksim	<code>\thicksim</code>
$\approx\!\!\!$	<code>\thickapprox</code>	\approxeq	<code>\approxeq</code>
\succcurlyeq	<code>\succapprox</code>	\precapprox	<code>\precapprox</code>
\backepsilon	<code>\backepsilon</code>		

Несколько символов из этой таблицы нам уже знакомы: в первую очередь это знаки для нестрогих неравенств \leq и \geq в привычном отечественному читателю начертании, а также знаки \sqsubset и \sqsupset , доступ к которым открывается уже при подключении пакета `latexsym`. Символы \triangleleft , \trianglelefteq , \triangleright и \trianglerighteq , задаваемые командами `\vartriangleleft`, `\trianglelefteq`, `\vartriangleright` и `\trianglerighteq`, также доступны уже при подключении пакета `latexsym`, но там они называются иначе: `\lhd`, `\unlhd`, `\rhd` и `\unrhd` соответственно.

Специальные команды предусмотрены для отрицаний отношений из предыдущей таблицы. В принципе, как мы помним, «отрицание» (перечеркнутый символ) можно напечатать, поставив перед этим символом команду `\not`, но взаимное расположение черты и перечеркиваемого символа при этом не всегда удачно. Поэтому Американское Математическое общество выделило для перечеркнутых символов специальные литеры (ради красоты приходится страдать ...). Итак:

$\not\leq$	<code>\nleq</code>	$\not\geq$	<code>\ngeq</code>	$\not\lessdot$	<code>\nless</code>
$\not\ngtr$	<code>\ngtr</code>	$\not\nprec$	<code>\nprec</code>	$\not\nsucc$	<code>\nsucc</code>
$\not\leqslant$	<code>\leqslant</code>	$\not\geqslant$	<code>\geqslant</code>	$\not\preceq$	<code>\preceq</code>
$\not\succcurlyeq$	<code>\succcurlyeq</code>	$\not\leqq$	<code>\leqq</code>	$\not\geqq$	<code>\geqq</code>
$\not\sim$	<code>\sim</code>	$\not\cong$	<code>\cong</code>	$\not\subsetneqq$	<code>\subsetneqq</code>
$\not\supseteqq$	<code>\supseteqq</code>	$\not\subseteqq$	<code>\subseteqq</code>	$\not\supseteq$	<code>\supseteq</code>
$\not\parallel$	<code>\parallel</code>	$\not\mid$	<code>\mid</code>	$\not\shortmid$	<code>\shortmid</code>
$\not\shortparallel$	<code>\shortparallel</code>	$\not\dashv$	<code>\dashv</code>	$\not\Vdash$	<code>\Vdash</code>
$\not\nVdash$	<code>\nVdash</code>	$\not\nVDash$	<code>\nVDash</code>	$\not\trianglerighteq$	<code>\trianglerighteq</code>
$\not\trianglelefteq$	<code>\trianglelefteq</code>	$\not\triangleleft$	<code>\triangleleft</code>	$\not\triangleright$	<code>\triangleright</code>

В следующей таблице мы собрали всевозможные стрелки (с точки зрения `TeX`'а, стрелки — это тоже знаки бинарных отношений, но математики, как правило, так не считают).

\rightsquigarrow	<code>\rightsquigarrow</code>	\circlearrowright	<code>\circlearrowright</code>
\circlearrowleft	<code>\circlearrowleft</code>	\leftrightharpoons	<code>\leftrightharpoons</code>
\rightleftharpoons	<code>\rightleftharpoons</code>	\twoheadrightarrow	<code>\twoheadrightarrow</code>
\twoheadleftarrow	<code>\twoheadleftarrow</code>	\leftleftarrows	<code>\leftleftarrows</code>
\rightrightarrow	<code>\rightrightarrow</code>	\upuparrows	<code>\upuparrows</code>
\downdownarrows	<code>\downdownarrows</code>	\upharpoonright	<code>\upharpoonright</code>

\downharpoonright	\upharpoonleft
\downharpoonleft	\rightarrowtail
\leftarrowtail	\leftrightarrowtail
\rightleftarrows	\downharpoonright
\upharpoonleft	\downharpoonleft
\rightarrowtail	\leftarrowtail
\leftrightarrows	\rightleftarrows
\Rightarrow \Rrightarrow	\Leftarrow \Lleftarrow
\nleftarrow	\nrightarrow
\nLeftarrow	\nRightarrow
\nLeftrightarrow	\nleftrightarrow
\curvearrowleft	\curvearrowright
\Lsh	\Rsh
\dasharrow	\dashleftarrow

Стрелка \rightsquigarrow доступна и при использовании пакета `latextsym`, но там она называется `\leadsto`.

Внимательный читатель мог заметить в приведенной выше таблице команду `\rightleftharpoons`, входящую в основной набор И^ТЕХ'а; ниже можно найти еще несколько аналогичных примеров. Такое дублирование — не прихоть ТЕХнологов из AMS: в базовом И^ТЕХ'е символы, задаваемые этими дублирующимися командами, собирались из отдельных кусочков, вследствие чего они не меняли должным образом размеры при помещении в индексы. В пакете `amssymb` те же команды отсылают к специальным литерам, входящим в шрифты AMS, в результате чего символы $=$ (`\rightleftharpoons`) или, скажем, \hbar (`\hbar`) правильно ведут себя и в индексах.

Теперь перечислим обычные символы, доступ к которым открывается при подключении пакета `amssymb`:

◊ \mho	□ \square
◊ \lozenge	■ \blacksquare
◆ \blacklozenge	★ \bigstar
▼ \blacktriangledown	▲ \blacktriangle
▽ \triangledown	∠ \measuredangle
△ \sphericalangle	(S) \circledS
□ \complement	∅ \varnothing
♯ \nexists	⋮ \Finv
○ \Game	ð \eth
□ \beth] \gimel
Γ \daleth	F \digamma
κ \varkappa	k \Bbbk
ℏ \hslash	¥ \yen

✓ \checkmark
 ✗ \maltese

® \circledR

Из этого набора нам знакомы символ \checkmark , доступ к которому открывает уже пакет *latexsym*, а также греческая буква \times и обозначение для пустого множества \emptyset . Символы \square и \diamond также доступны уже при подключении пакета *latexsym*, но там они называются \Box и \Diamond соответственно.

Команды \yen, \checkmark, \circledR и \maltese можно использовать не только в формулах, но и в тексте (кстати, ¥ — это обозначение для японской иены).

И, наконец, стилевой пакет *amsymb* определяет четыре дополнительных «ограничителя» (напомним, что ограничители — это символы, выступающие в роли скобок):

⌜ \ulcorner ⌝ \urcorner ⌞ \llcorner ⌠ \lrcorner

Наш перечень символов, определяемых в стилевом пакете *amsymb*, завершен. Осталось сказать еще о двух вещах. Во-первых, некоторые из определенных выше символов становятся доступными уже при использовании *amsfonts*. Их перечень таков:

¥ \yen	✓ \checkmark
® \circledR	✗ \maltese
--> \dasharrow	--> \dashleftarrow
⊓ \sqsubset	⊔ \sqsupset
⊑ \vartriangleleft	⊒ \vartriangleright
⊑ \trianglelefteq	⊒ \trianglerighteq
⊓ \square	◊ \lozenge
~~ \rightsquigarrow	= \rightleftharpoons

Поскольку пакет *amsymb* довольно громоздок, стоит иметь в виду возможность иногда обойтись более скромным *amsfonts*.

Во-вторых, у некоторых из перечисленных команд есть синонимы. Вот их список:

--> \dasharrow	или	\dashrightarrow
÷ \Doteq	или	\doteqdot
⊓ \Cup	или	\doublecup
⊑ \Cap	или	\doublecap
≪ \lll	или	\llless
≫ \ggg	или	\gggtr

Приложение Е

AMS-LATEX в рамках *LATEX'a 2_ε*

Мы уже неоднократно упоминали об *AMS-LATEX*'е, сочетающем мощь *AMS-TEX*'а и удобство *LATEX'a*. Давайте познакомимся с ним поближе.

Современный *AMS-LATEX* есть не что иное, как совокупность связанных друг с другом стилевых пакетов и классов документов для *LATEX'a 2_ε*, воспроизводящих в рамках *LATEX'a 2_ε* практически все возможности *AMS-TEX*'а как по набору формул, так и по оформлению документов в соответствии со стандартом Американского математического общества. Основные *AMS-LATEX*'овские средства набора формул содержатся в стилевых пакетах *amsmath* и *amscd*. Дополнительные шрифты, предоставляемые Американским математическим обществом для набора формул, содержатся в пакетах *amsfonts* (готический и ажурный шрифты плюс несколько спецзнаков) и *amssymb* (все то, что дает *amsfonts*, плюс основная масса новых спецзнаков)³³. Дополнительные возможности работы с окружениями типа «теорема» содержатся в пакете *amsthm*. Наконец, оформление документа по стандарту AMS предусмотрено в классах документов *amsart*, *amsproc* и *amsbook* (при использовании этими классами автоматически подключаются *amsmath* и *amsthm*).

До появления *LATEX'a 2_ε* под *AMS-LATEX*'ом понимали набор стилевых файлов для так называемого «*LATEX'a с NFSS» — версии, промежуточной между *LATEX'ом 2.09* и *LATEX'ом 2_ε*. Этот «старый» *AMS-LATEX*, описанный в книге [8], называют еще версией 1.1 (а тот, о котором мы рассказываем, — версией 1.2).*

Кое с чем из *AMS-LATEX'a* вы уже знакомы. Так, в гл. II мы рас-

³³Чтобы пользоваться дополнительными шрифтами, доступ к которым открывается в двух последних пакетах, необходимо иметь и эти шрифты как таковые!

сказывали о пакете *amsfonts*, позволяющем использовать в формулах готический и ажурный шрифты. В той же главе мы рассказали и о пакете *amscd*, помогающем при наборе коммутативных диаграмм, а в приложении Д шла речь о пакете *amssymb*. Сейчас мы расскажем и об остальных средствах *AMS-LATEX*'а (опустив некоторые детали).

Многое из дальнейшего будет хорошо знакомо любителям *AMS-TeX*'а, хотя обозначения будут порой не такими, как в книге [6].

1. Матрицы (пакет amsmath)

Все возможности, о которых идет речь в этом разделе, доступны пользователю L^AT_EX'a 2_E при подключении стилевого пакета *amsmath* (иногда мы будем опускать эту оговорку).

Для набора матриц, заключенных в круглые скобки, стилевой пакет `amsmath` предоставляет окружение `pmatrix`. Вот как оно работает:

```


$$\begin{pmatrix} a_{11} - \lambda & a_{12} & a_{13} \\ a_{21} & a_{22} - \lambda & a_{23} \\ a_{31} & a_{32} & a_{33} - \lambda \end{pmatrix} \quad \begin{aligned} &\$ \$\$ \begin{pmatrix} a_{11} - \lambda & a_{12} & a_{13} \\ a_{21} & a_{22} - \lambda & a_{23} \\ a_{31} & a_{32} & a_{33} - \lambda \end{pmatrix} \\ &\$ \$\$ \end{aligned}$$


```

В отличие от окружения `array`, здесь не нужно указывать `\TeX`'у, сколько и каких столбцов будет в матрице (число столбцов не должно превышать десяти); скобки получаются автоматически.

Прямоугольные таблицы из формул бывают заключены не только в круглые скобки; соответственно, определены окружения `bmatrix`, `vmatrix` и `Vmatrix`, отличающиеся от `pmatrix` только тем, что вместо круглых скобок таблица заключена соответственно в квадратные скобки [], вертикальные черточки || и удвоенные вертикальные черточки |||. Есть также окружение `matrix`, которое дает на печати только прямоугольную таблицу, без всяких скобок (и с меньшими интервалами между столбцами, чем в окружении `array`).

Чтобы получить в матрице горизонтальный ряд точек, простирающийся на несколько столбцов, используется команда `\hdotsfor`; ее обязательный аргумент — количество столбцов, занятых точками. В приведенном ниже примере обратите внимание на расстановку знаков & в строке, содержащей `\hdotsfor`:

0	0	a_1
1	0	a_2
.....			
....	1	0	a_{n-1}
0	1	a_n

```
 $$\begin{vmatrix}
 0 & 0 & \cdots & a_1 \\
 1 & 0 & \cdots & a_2 \\
 \cdots & \cdots & \cdots & \cdots \\
 \cdots & 1 & 0 & a_{n-1} \\
 0 & \cdots & 1 & a_n
 \end{vmatrix}$$
```

Можно также регулировать густоту точек, получаемых при помощи команды `\hdotsfor`: в необязательном аргументе (он ставится перед обязательным) можно указать десятичную дробь — «коэффициент разреживания». Если сказать `\hdotsfor[1.5]{5}` вместо `\hdotsfor{5}`, то точки будут идти в полтора раза реже.

Если вам понадобятся матрицы с более чем десятью столбцами, можно изменить максимальное количество столбцов, присвоив в преамбуле документа соответствующее значение счетчику `MaxMatrixCols`.

2. Маленькие хитрости (пакет *amsmath*)

Все, о чем идет речь в этом разделе, также становится доступным при подключении пакета *amsmath*.

Для печати в формуле прописных греческих букв в наклонном начертании применяются специальные команды, в которых перед названием буквы стоит `var`:

Γ	<code>\varGamma</code>	Δ	<code>\varDelta</code>	Θ	<code>\varTheta</code>
Λ	<code>\varLambda</code>	Ξ	<code>\varXi</code>	Π	<code>\varPi</code>
Σ	<code>\varSigma</code>	Υ	<code>\varUpsilon</code>	Φ	<code>\varPhi</code>
Ψ	<code>\varPsi</code>	Ω	<code>\varOmega</code>		

Команда `\mathit` для получения наклонных прописных греческих букв при подключении пакета *amsmath* не сработает.

Для включения текста в формулу лучше пользоваться не командой `\mbox`, а командой `\text`. При этом включенный в формулу текст будет правильно менять размеры в степенях и индексах; если же включенный в формулу текст сам, в свою очередь, содержит формулу, то размеры символов в этой формуле будут выбраны более правильно, чем при использовании `\mbox`.

При подключении пакета *amsmath* команда `\bigl` и ей подобные, предназначенные для явного указания размера «ограничителей», будут правильно работать и со шрифтами кеглей 11 и 12 (ср. с. 62).

Многоточие в формуле можно задавать командой `\dots`: при этом \TeX автоматически выберет (в соответствии с американскими полиграфическими традициями), ставить его внизу строки или по центру. Если многоточие попадает в конец формулы, то надо все же явно указать `\ldots` или `\cdots`.

Для ссылки на математическую формулу можно вместо `\ref` использовать команду `\eqref`, которая автоматически ставит скобки вокруг номера.

Чтобы нумерация формул шла не сплошняком (как будет в стиле `article`), а начиналась заново в каждом разделе, определенном командой `\section`, можно использовать команду `\numberwithin` так:

```
\numberwithin{equation}{section}
```

В общем случае, если `abcd` и `defg` — два счетчика, то команда

```
\numberwithin{abcd}{defg}
```

делает счетчик `abcd` подчиненным счетчику `defg`, и при этом переопределяет печатное представление счетчика `abcd`, добавляя в начало печатное представление `defg`, а затем точку. В начале гл. IX мы объясняли, как делать такие вещи вручную.

Команды `\xleftarrow` и `\xrightarrow` предназначены для нанесения надписей над и под стрелками. В обязательном аргументе этих команд ставится надпись над стрелкой, в необязательном — под стрелкой. Если надпись длинная, размер стрелки автоматически увеличивается:

$$A \xleftarrow[z]{f} B \xrightarrow[f+g-h]{z} C \quad \text{---} \quad \begin{aligned} & \$\$ A \xleftarrow[z]{f} B \\ & \xrightarrow[f+g-h]{z} C \$\$ \end{aligned}$$

Для набора цепных дробей имеется специальная команда `\cfrac`. Как ею пользоваться, должно быть ясно из следующего примера:

$$\frac{7}{25} = \cfrac{1}{3 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{3}}}} \quad \begin{aligned} & \$\$ \cfrac{7}{25} = \\ & \cfrac{1}{3 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{3}}}} \\ & \$\$ \end{aligned}$$

В $\text{\TeX}'e$ 2.09 грамотный набор цепной дроби требовал известной изощренности (с. 78). Кстати, если вы хотите, чтоб какой-то из числителей в цепной дроби был центрирован, а выключен влево или вправо, надо вместо `\cfrac` сказать `\cfrac[1]` или `\cfrac[r]` соответственно.

Для расстановки в формулах акцентов над буквами в пакете *amsmath* определены команды `\hat`, `\check`, `\tilde`, `\acute`, `\grave`, `\dot`, `\ddot`, `\breve`, `\bar` и `\vec`. От аналогичных *TEX*'овских команд с именами, начинающимися со строчных букв (с. 63), они отличаются тем, что правильно ставят двойные акценты:

Правильно \hat{Z} , Правильно $\hat{\hat{Z}}$, а не $\hat{\hat{Z}}$.

Наряду с командами `\pmod` и `\bmod`, в пакете *amsmath* определены их аналоги `\mod` и `\pod`, обозначающие то же понятие, что `\mod`, другими способами:

$$\begin{aligned} a^{p-1} \equiv 1 \pmod p \\ a^{p-1} \equiv 1 \pmod p \end{aligned} \quad \begin{aligned} \$a^{p-1} \equiv 1 \pmod p \\ \$a^{p-1} \equiv 1 \pmod p \end{aligned}$$

Определены команды `\varlimsup` и `\varliminf`, дающие альтернативные обозначения для верхнего и нижнего пределов, а также команды `\varinjlim` и `\varprojlim`, дающие альтернативные обозначения для индуктивного и проективного пределов:

$$\begin{aligned} \overline{\lim}_{n \rightarrow \infty} a_n = \inf_n \sup_{m \geq n} a_m \\ \mathcal{F}_x = \varinjlim_{U \ni x} \mathcal{F}(U) \end{aligned} \quad \begin{aligned} \$\varlimsup_{n \rightarrow \infty} a_n \\ \$a_n = \inf_n \sup_{m \geq n} a_m \\ \$\mathcal{F}_x = \\ \$\varinjlim_{U \ni x} \mathcal{F}(U) \end{aligned}$$

Для создания новой команды для «математической операции» типа `sin` в пакете *amsmath* есть команда `\DeclareMathOperator`. Если вам нужно определить команду `\Tor`, дающую в формуле `Tor` с приличествующими случаю пробелами, достаточно написать

```
\DeclareMathOperator{\Tor}{Tor}
```

В первом аргументе ставится придуманное вами имя команды (незанятое, естественно), во втором — то, что вы хотите получить на печати. Содержимое второго аргумента будет обработано в математическом режиме, но при этом символы `-` (дефис), `*` и `'` будут иметь такое же значение, как в обычном тексте (это удобно, если вы хотите, чтобы имя вашего нового оператора включало тот же дефис). Есть и команда `\DeclareMathOperator*`; при «операторе», определенном такой командой, «пределы» будут ставиться так же, как при `lim`.

Определены команды `\iint`, `\iiint` и `\iiiiint` для двойного, тройного и «четверного» интегралов (если просто написать несколько подряд команд `\int`, то получатся слишком большие пробелы между знаками интеграла; см. с. 77).

Можно взять формулу в рамочку с помощью команды `\boxed{}`.

Вот пример к двум последним абзацам (в этом примере мы подразумеваем, что подключен и пакет `amsfonts` или `amssymb`):

$$\iint_{\mathbb{R}^2} e^{-(x^2+y^2)} dx dy = \pi$$

```
$$\boxed{\int\limits_{\mathbb{R}^2} e^{-(x^2+y^2)} dx dy = \pi}$$
```

3. Дроби и их обобщения (пакет `amsmath`)

Еще раз напоминаем: то, о чем пойдет речь, доступно в $\text{\TeX}'a 2\varepsilon$ при подключении пакета `amsmath`.

Начнем с дробей, набираемых с помощью команды `\frac`. Если вы считаете, что \TeX выбрал для числителя и знаменателя такой дроби слишком мелкий шрифт (например, когда дробь сама входит в числитель или знаменатель другой дроби), то к вашим услугам команды `\tfrac` (дробь в «текстовом» стиле — `textstyle`, см. с. 77) и `\dfrac` (дробь в «выключном» стиле — `displaystyle`)

$$\frac{x^2}{y^2 + \frac{a^2}{b^2}} \qquad \frac{x^2}{y^2 + \frac{a^2}{b^2}} \qquad \begin{aligned} & \quad \frac{2}{\dfrac{xy}{ab} + \dfrac{ab}{xy}} \\ & \quad \qquad \qquad \qquad \end{aligned}$$

Как и в случае с обычной `\frac`, можно не брать в фигурные скобки числитель и/или знаменатель, состоящий из одной буквы.

При подключении пакета `amsmath` формулы типа $\binom{a}{b}$ или $\binom{i}{k}$ набираются не с помощью $\text{\TeX}'$ овских команд `\choose` или `\atop` (более того, применять их в *AMS- \TeX* 'е и нельзя), а с помощью специальных конструкций. Для формул типа $\binom{a}{b}$ предусмотрена специальная команда `\binom`, работающая аналогично `\frac`; есть также ее варианты `\tbinom` и `\dbinom`.

Возможности отмененной в *AMS- \TeX* 'е команды `\atop` (а также нечто большее) реализуются, при подключении пакета `amsmath`, в концепции «обобщенной дроби». По определению, обобщенная дробь — это фрагмент формулы, устроенный так: левый ограничитель, затем дробь (толщина дробной черты может быть произвольной, в том числе нулевой), затем правый ограничитель. Напомним, что ограничители — это скобки и им подобные символы, способные автоматически менять размер (с. 60); в обобщенной дроби ограничители могут и отсутствовать (так что обычная дробь — действительно частный случай обобщенной). Для набора обобщенной дроби предусмотрена команда `\genfrac` с шестью аргументами. Чтобы понять, как она работает, посмотрим на пример:

Формула $\left[\frac{x}{y-z} \right]$ лишена всякого смысла.

Формула
 $\$\\genfrac{(){}{}{1pt}{0}{x}{y-z}\\$$
 лишена всякого смысла.

Первый и второй аргументы команды \genfrac — это левый и правый ограничители соответственно; третий аргумент — толщина дробной черты (если толщина нулевая, то дробная черта не печатается); четвертый аргумент содержит указания по поводу размера шрифта для числителя и знаменателя (если оставить его пустым, написав просто $\{\}$ вместо $\{0\}$, то \TeX выберет размер самостоятельно; цифра 0 означает *displaystyle*, цифра 1 — *textstyle*); наконец, пятый и шестой аргументы — это собственно числитель и знаменатель.

Если оставить третий аргумент пустым, написав просто $\{\}$ вместо фигурных скобок, в которых записана толщина, то будет выбрана толщина дробной черты по умолчанию (она равна 0.4 пункта). Если оставить первый и второй аргумент пустыми, то ограничителей не будет (если, однако, левый ограничитель указан, то должен быть указан и правый!). Например, $\text{\frac}{x}{y}$ — это то же самое, что

$\text{\genfrac}{(){}{}{}{x}{y}}$

Конечно, команда \genfrac хороша не сама по себе, а как сырье для определения макросов, приспособленных к вашим конкретным нуждам.

4. Выключные формулы (пакет *amsmath*)

По сравнению с довольно скучными средствами для набора многострочных выключных формул, предоставляемыми обычным $\text{\TeX}'ом$, пакет *amsmath* предоставляет массу возможностей.

Если выключная формула столь длинна, что не умещается в строке, то к вашим услугам окружение *multiline*:

$$\begin{aligned} 1 + 2 + 3 + 4 + \cdots \\ + 46 + 47 + 48 + \cdots \\ + 99 + 100 = 5050 \quad (1) \end{aligned}$$

```
\begin{multiline}
1+2+3+4+\cdots \\
+46+47+48+\cdots \\
+99+100=5050
\end{multiline}
```

Первая из строк печатается выключенной влево, последняя — выключенной вправо, остальные строки центрируются. Подобно окружению *equation*, окружение *multiline* не должно быть заключено в знаки $\$$. Как вы могли заметить, формула, оформленная в виде окружения *multiline*, автоматически нумеруется. Чтобы этой нумерации не было, надо воспользоваться «вариантом со звездочкой» — окружением *multiline**.

На самом деле первая и последняя строки печатаются не вплотную к полям, а с отступом, равным `\multlinegap`. Значение этого параметра можно изменить с помощью `\setlength`.

Чтобы какая-то из средних строк была не центрирована, а выключена влево, надо воспользоваться командой `\shoveleft`, написав, скажем,

```
\showleft{+46+47+48+\cdots}\%
```

вместо `+46+47+48+\cdots\|.` Для выключки вправо аналогичным образом используется команда `\shoveright`.

Когда несколько выключных формул идут подряд, нехорошо оформлять каждую из них с помощью \$\$, так как вертикальные интервалы между соседними формулами будут слишком велики. Лучше воспользоваться окружением `gather`:

```


$$2 \times 2 = 4 \quad (2) \quad \begin{gathered} \\ \end{gathered}$$


$$24 \times 24 = 576 \quad (3) \quad \begin{gathered} \\ \end{gathered}$$


```

При использовании *gather* формулы также не должны быть заключены в символы **$\$$** . Каждая из формул, собранных в *gather*, автоматически нумеруется. Если какую-то из них нумеровать не надо, следует поставить непосредственно перед \\\ команду *\notag*. Если вы не хотите нумеровать ни одну из формул, можно воспользоваться «вариантом со звездочкой» — окружением *gather**.

При использовании `gather` все формулы будут центрированы. Иногда предпочтительнее было бы расположить их с выравниванием. Кое-что в этом плане дает знакомое вам окружение `eqnarray`, но если уж вы подключили `amsmath`, то предпочтительнее пользоваться окружением `align`, позволяющим напечатать не один, а несколько выровненных столбцов формул:

$$\begin{array}{lll} 7 \times 9 = 63 & 63 : 9 = 7 & (4) \\ 8 \times 9 = 72 & 72 : 9 = 8 & (5) \end{array} \quad \begin{aligned} & \text{\begin{align}} \\ & 7\text{\times }9\text{\&}=63 \text{ \& } 63\text{:9\&}=7\\ & 8\text{\times }9\text{\&}=72 \text{ \& } 72\text{:9\&}=8 \\ & \text{\end{align}} \end{aligned}$$

Обратите внимание, что знак **&** ставится только перед знаком равенства, в отличие от того, что мы имели в окружении **eqnarray**. Второй знак **&** в строке отделяет первый столбец формул от второго, по третьему знаку **&** идет выравнивание во втором столбце, четвертый **&**, если бы он был, отделял бы второй столбец от третьего, и т. д. По-прежнему не нужны знаки **\$\$**, каждое уравнение автоматически получает номер,

который можно подавить, написав `\notag` перед `\backslash`, и по-прежнему есть вариант со звездочкой `, который формулы не нумерует.`

При грамотном применении окружения `в строке должно стоять нечетное число знаков &. Именно, если у нас n столбцов с уравнениями, то имеется $n - 1$ знаков &, отделяющих друг от друга столбцы, плюс еще n знаков — по одному на каждый столбец, а всего $(n - 1) + n = 2n - 1$.`

Полезное применение `возникает, когда идущие подряд выключные формулы содержат текстовые комментарии. Желательно, чтобы эти комментарии были выровнены. Вот как можно этого добиться с помощью :`

$$3 \cdot 5 + 7 \cdot 5 = (3 + 7) \cdot 5 \quad (\text{ясно}) \\ = 50 \quad (\text{очевидно})$$

```
\begin{aligned}
3 \cdot 5 + 7 \cdot 5 &= (3 + 7) \\
&= 50
\end{aligned}
```

`\begin{aligned}`
`3 \cdot 5 + 7 \cdot 5 &= (3 + 7)`
`\cdot 5 && \text{(ясно)} \\`
`\cdot 7 && \text{(очевидно)}`
`\end{aligned}`

Обратите внимание на два амперсенда, отделяющие комментарий от формул (см. выше текст мелким шрифтом). Приятно также отметить, что при пользовании `получаются правильные пробелы вокруг знака равенства (ср. с. 69).`

Не всегда удобно включать комментарии к выкладкам прямо в формулы. Иногда хочется, чтобы какой-то из комментариев шел в отдельной строке. Команда `\intertext` позволяет сделать это так, чтобы выравнивание не нарушилось:

$$3 \cdot 5 + 7 \cdot 5 = (3 + 7) \cdot 5 \quad (\text{ясно}) \\ = 50 \quad (\text{очевидно}),$$

откуда

$$15 + 35 = 50$$

```
\begin{aligned}
3 \cdot 5 + 7 \cdot 5 &= (3 + 7) \\
&= 50
\end{aligned}
```

`\intertext{откуда}`

$$15 + 35 = 50$$

```
\begin{aligned}
15 + 35 &= 50
\end{aligned}
```

Наряду с окружением `, дающим сразу целую выключную формулу, есть окружение , которое можно использовать в качестве составной части большей формулы. Вот как можно с помощью этого окружения задать систему уравнений:`

$$\left\{ \begin{aligned} x^2 + y^2 &= 7 \\ x + y &= 3. \end{aligned} \right.$$

```

$$
\left\{ \begin{aligned}
&x^2+y^2=7\\
&x+y = 3.\end{aligned} \right.
\right.

```

Сравните с тем, что получается при использовании обычного *LATEX'овского* окружения *array* (с. 69).

5. Работа с теоремами (пакет *amsthm*)

Все *LATEX'овские* «теоремы», определяемые пользователем при помощи окружения *newtheorem* (см. с. 239 и далее), оформляются в одном и том же стиле, что не всегда приемлемо. Пакет *amsthm* позволяет внести в это оформление некоторое разнообразие.

Во-первых, в этом пакете определен «вариант со звездочкой» команды *\newtheorem*. Именно, если определить очередной тип «теорем» с помощью *\newtheorem** вместо *\newtheorem*, то «теоремы» указанного типа не будут нумероваться.

Во-вторых, для управления стилем оформления окружений типа «теорема» предназначена команда *\theoremstyle*. Это — команда с одним аргументом. В качестве ее аргумента можно использовать слова *plain*, *definition* или *remark*. Если в преамбуле дать эту команду с одним из трех допустимых аргументов, то все «теоремы», определяемые с помощью *\newtheorem* после этой команды *\theoremstyle*, будут оформлены в соответствующем стиле; чтобы определить тип теорем, оформляемый в другом стиле, надо написать еще одну команду *\theoremstyle* (с другим аргументом, надо думать), а уж после нее — очередную *\newtheorem*. Стиль *plain* рекомендуется для собственно теорем, *definition* — для определений, *remark* — для замечаний. Если в преамбуле нет ни одной команды *\theoremstyle*, подразумевается стиль *plain*.

Пакет *amsthm* предусматривает также окружение *proof*, предназначенное для оформления доказательств. Это окружение автоматически ставит слово *Proof* в начало доказательства, и автоматически же завершает доказательство символом \square . Если вас не устраивает, что слово «доказательство» пишется по-английски, нужно переопределить с помощью *\renewcommand* команду *\proofname* (ср. с. 154).

Окружение `proof` допускает и необязательный аргумент: если написать, скажем,

```
\begin{Proof}[Доказательство основной теоремы]}
```

то вместо слова `Proof` появится текст, записанный нами в квадратных скобках.

6. *AMS-TEX*'овские классы документов

В состав *AMS-TEX*'а входят три класса документов: `amsart`, `amsproc` и `amsbook`. Оставляя последний в стороне (уж если Американское математическое общество закажет вам монографию, то, наверное, снабдит и подробными инструкциями), остановимся на особенностях оформления документа «в целом», характерных для первых двух классов.

Для начала напомним, что *AMS-TEX*'овские классы документов автоматически подключают стилевые пакеты `amsmath` и `amsthm` (если, однако, вам понадобились коммутативные диаграммы или дополнительные шрифты, то пакет `amscd`, `amsfonts` или `amssymb` вам все же придется подключить в явном виде, с помощью команды `\usepackage`). Остальные особенности *AMS-TEX*'овских классов относятся к титульной информации (т. е. тому, что идет до команды `\maketitle`) и рубрикации документа.

Начнем с титульной информации. Команды `\title` и `\author` могут принимать необязательный аргумент (в квадратных скобках, естественно), ставящийся перед обязательным. Эти необязательные аргументы суть сокращенные варианты заглавия и имени автора, предназначенные для включения в колонтитулы. Команда `\thanks`, в аргументе которой обычно выражается благодарность за финансовую поддержку, оформляется не как сноска к имени автора, а записывается в преамбулу самостоятельно, наравне с `\author` и `\title`. В преамбуле одного документа может быть несколько команд `\thanks`. Команда `\address` также принимает один обязательный аргумент — адрес автора (можно разбивать на строки командой `\`). Если вы хотите, наряду с постоянным адресом, указать еще и адрес для текущей переписки, можно это сделать в аргументе команды `\curraddr`. А если автор имеет еще и электронный адрес, можно его указать в команде `\email` (вниманию привыкших к *TeX*'у: символ `@` надо набирать непосредственно, не используя `backslash` и/или удвоения!). Команда `\address` должна следовать после команды `\author`; если вы пользуетесь командами `\curraddr` и/или `\email`, то они, в свою очередь, должны следовать после команды `\address` именно в таком порядке, как указано выше.

Все сказанное относилось к случаю, когда автор у документа один. Если авторов несколько, то информацию о каждом из них надо задавать *отдельной* командой `\author`; после каждой команды `\author` ставится своя команда `\address`, а также, если нужно, `\curraddr` и/или `\email`.

Если вы хотите посвятить кому-то свою работу, запишите это посвящение в аргументе команды `\dedicator`. В аргументе команды `\keywords` вы можете записать список ключевых слов, определяющих вкратце, что надо знать для понимания вашего труда, а в аргументе команды `\subjclass` — указать, к какому разделу математики, согласно рубрикатору AMS, относится ваш опус. Наконец, аннотация к статье, то есть окружение `abstract`, в *AMS-ЛАТЭХ*'е должна идти до `\maketitle`.

Что касается самого текста статьи, то главное отличие от того, что нам привычно по «обычному» ЛАТЭХ'у, состоит в командах рубрикации. Классы `amsart` и `amsproc` допускают в качестве таковых только `\section`, `\subsection`, `\subsubsection`, а также команду `\specialsection`, задающую самые крупные рубрики (более крупные, чем `\section`).

Теперь вы знаете достаточно про *AMS-ЛАТЭХ*, чтобы оформить статью по математике в соответствии с канонами Американского математического общества. Осталось только написать эту статью. Успехов вам!

Приложение Ж

Откуда взять TeX?

Если вы прониклись мыслью, что L^AT_EX — это хорошо, то вам, наверное, захочется узнать, как его достать. Настоящее приложение посвящено ответу на этот вопрос.

Любая издательская система на базе TeX'a (L^AT_EX в том числе) состоит из следующих компонент.

Транслятор (точнее, интерпретатор) языка TeX. Это исполняемый файл (для компьютеров, работающих под системой DOS, — *exe*-файл), читающий ваш исходный текст и превращающий его в *dvi*-файл.

dvi-драйверы. Это также исполняемые файлы, предназначенные для просмотра на экране, печати, и т. п.

Шрифты. Это набор *tfm*-файлов (они содержат информацию о размерах символов) и *pk*-файлов (они содержат информацию о форме символов) — см. приложение А.

Исходные тексты. Это текстовые файлы, написанные на языке TeX и содержащие определения всех команд, используемых в данном макропакете.

Интерпретаторы и *dvi*-драйверы для каждого типа компьютеров свои (интерпретатор TeX'a для IBM PC на Макинтош не перенесешь), а исходные тексты и шрифты переносимы с платформы на платформу.

Один и тот же интерпретатор TeX'a может работать с различными макропакетами: L^AT_EX, A_MS_TE_X, Plain TeX и т. п. Начинающие часто задают вопрос наподобие: «Что лучше, емTeX или A_MS_TE_X?». Этот вопрос лишен смысла: такой интерпретатор TeX'a, как емTeX, может

работать не только с *AMSTeX*'ом, но и с другими TeX'овскими макропакетами (скажем, с *LATEX*'ом), а макропакет *AMSTeX* можно использовать не только с *emTeX*'ом, но и с другими интерпретаторами (скажем, с *PCTeX*'ом).

Интерпретаторы TeX'a и dvi-драйверы являются, как правило, свободно распространяемыми программными продуктами (хотя также существуют shareware и коммерческие версии, как например упомянутый выше *PCTeX*). Все известные автору исходные тексты макропакетов также распространяются свободно. То обстоятельство, что программный продукт распространяется свободно, избавляет пользователя от необходимости платить за него (или от моральных мук по поводу использования нелегальной копии), но не от определенных обязательств: свободно распространяемые продукты считается неэтичным продавать, распространять же их можно только в виде полного комплекта, а не отдельных составных частей. Наконец, в такие продукты нельзя вносить изменения, кроме специально оговоренных мест. Точная информация по поводу того, что можно и что нельзя делать со свободно распространяемым программным продуктом, поставляется обычно вместе с ним.

Где же, однако, взять эти свободно распространяемые продукты? Сначала мы ответим на этот вопрос применительно к «родному» TeX'u, рассчитанному на латинский алфавит, а затем объясним, как обстоит дело с русификацией.

1. Электронный дом для TeX'a

Начнем с наиболее естественного (пока, к сожалению не самого доступного для отечественного читателя) способа. Если вы имеете доступ в Internet, то любой свободно распространяемый программный продукт, имеющий отношение к TeX'u, можно получить по анонимному ftp по адресу *ftp.dante.de* в директории */pub/tex* (или */tex-archive*). На этом ftp-сервере расположен полный TeX'овский архив, именуемый CTAN (Comprehensive TeX Archive Network). Тот же архив расположен по адресам *ftp.shsu.edu* и *ftp.tex.ac.uk*. Общий объем информации, содержащейся в этом архиве, составляет около гигабайта (вам из этого потребуется от силы несколько десятков мегабайт).

Архив CTAN устроен следующим образом. Интерпретаторы TeX'a и dvi-драйверы собраны в поддиректории *systems*, исходные тексты макропакетов содержатся в поддиректории *macros*, а в поддиректории *fonts* хранятся шрифты. Пользователям IBM PC разумно исполь-

зователь интерпретатор и *dvi*-драйверы, входящие в пакет *emTeX*. Они содержатся в директории

/tex-archive/systems/msdos/emtex

В состав пакета *emTeX*, наряду с собственно интерпретатором *TeX*'а и *dvi*-драйверами, входит много других полезных вещей, в частности:

- исходные тексты Plain *TeX*'а, *LATeX*'а 2.09 и *LATeX*'а 2 ε ;
- набор шрифтов, используемых Plain *TeX*'ом и *LATeX*'ом, вместе с их *METAFONT*'овскими исходными текстами;
- программа *makeindex* (см. приложение Г);
- интерпретатор *METAFONT*'а (*dvi*-драйверы устроены таким образом, что *METAFONT* автоматически генерирует по мере надобности недостающие *pk*-файлы с очертаниями тех шрифтов, исходные тексты которых присутствуют на вашем компьютере);
- краткое руководство по *LATeX*'у (на немецком языке).

Для работы с *LATeX*'ом 2 ε желательно также иметь шрифты Американского математического общества и *AMS-LATeX*'овские стилевые пакеты. Эти материалы в комплект поставки *emTeX*'а не входят; их можно взять в том же архиве в директориях

/tex-archive/fonts/ams/amsfonts

(шрифты вместе с файлами, необходимыми для подключения их к *LATeX*'у 2 ε) и

/tex-archive/fonts/ams/amsfonts/amslatex

(*AMS-LATeX*-овские стилевые пакеты и классы документов).

В обширной документации к пакету *emTeX* содержатся и подробные инструкции по его установке. Помимо распаковки *zip*-файлов и установки переменных окружения, существенным этапом является генерация так называемых форматных файлов, в которые записывается внутреннее представление многочисленных макроопределений, содержащихся в исходных текстах. При запуске *LATeX*'а на самом деле запускается интерпретатор *TeX*'а с указанием прочесть *LATeX*'овский форматный файл (это гораздо быстрее, чем интерпретация исходного текста). Форматные файлы для каждого макропакета и для каждой реализации *TeX*'а различны.

Наряду с форматными файлами, при установке *emTeX'a* необходимо создать и так называемые базовые файлы **METAFONT'a**, чтобы **METAFONT** смог при необходимости сгенерировать недостающие шрифты.

Исходные тексты для **LATeX'a** можно получить и независимо от какого-либо интерпретатора **TeX'a**. Исходные тексты **LATeX'a 2_ε** содержатся в директории

/tex-archive/macros/latex

(шрифты Американского математического общества и **AMS-LATeX'овские** стилевые пакеты опять-таки расположены в другом месте — см. выше). Исходные тексты **LATeX'a 2.09** хранятся в директории

/tex-archive/macros/latex209

2. Ассоциации пользователей **TeX'a** и проблемы русификации

Пользователь, не имеющий возможности получать файлы по **ftp**, может вступить в ассоциацию пользователей **TeX'a** (по-английски эти ассоциации называются **TeX Users Groups**) и получить **TeX'овский** комплект вместе с инструкциями по его установке (возможно, за небольшую плату, взимаемую в качестве членского взноса в ассоциацию или для компенсации расходов на копирование дисков). Такие ассоциации существуют во многих странах, в том числе и в России. Прежде чем говорить об отечественной ассоциации пользователей **TeX'a**, остановимся на общих проблемах, связанных с русификацией **TeX'a**.

Чтобы заставить **TeX** обрабатывать русские тексты, надо решить следующие проблемы:

- необходимы кириллические шрифты;
- для корректной обработки русских текстов необходимы некоторые дополнения и/или изменения к исходным текстам макропакетов (пример одной из проблем, которые приходится решать: **TeX** надо научить соответствуанию между прописными и строчными русскими буквами), а для **LATeX'a**, кроме того, необходимо написать специальный «русский» стилевой файл (стилевой пакет, если речь идет о **LATeX'e 2_ε**);
- необходимо дополнение к таблице переносов, чтобы **TeX** смог правильно переносить русские слова.

Третья из этих проблем была решена Дмитрием Вулисом (США): составленная им таблица русских переносов дает приемлемые результаты и используется в настоящее время всеми пользователями TeX'a в России. Первые две проблемы решались различными группами пользователей независимо друг от друга и немного по-разному. В результате в настоящее время в России имеют хождение несколько различных русификаций TeX'a; I^AT_EX'овский файл, подготовленный для одной из этих русификаций, можно после небольших переделок обработать с помощью другой русификации TeX'a. Ниже мы упомянем о нескольких группах разработчиков русификаций TeX'a. Этот список заведомо не является полным; автор приносит извинения тем, кого он не упомянул, и хочет подчеркнуть, что упоминание или неупоминание кого бы то ни было в настоящей книге не несет никакой оценочной нагрузки.

Практически все отечественные пользователи создавали кириллические шрифты, начертания которых имитируют начертания стандартных TeX'овских шрифтов Computer Modern. Первыми на этом пути были, видимо, Александр Самарин и Нана Глонти из Института физики высоких энергий в Протвино; в последнее время получили распространение шрифты, разработанные Ольгой Лапко (ассоциация *CyrTUG*). Автору известны два исключения: в русификации TeX'a, предпринятой Михаилом Виноградовым (Институт народнохозяйственного прогнозирования) присутствуют, наряду с русским аналогом Computer Modern, различные традиционные полиграфические шрифты (эта русификация, называемая *ViT_EX*, не является свободно распространяемым продуктом); кроме того, в русификации Александра Шения (Независимый Московский университет) предпринята попытка использовать (опять-таки наряду с русским аналогом Computer Modern) русский шрифт, аналогичный по начертанию шрифтам гарнитуры Таймс.

Что же касается изменений и дополнений к исходным текстам, то с точки зрения пользователя основные различия между русификациями сводятся к способу задания знака номера и кавычек-«елочек» и „лапок“ (при условии, конечно, что соответствующие знаки присутствуют в русских шрифтах), а также к оформлению первых строк в исходном файле (обычно требуется указать стилевую опцию, а в случае I^AT_EX'a 2_E — подключить стилевой пакет, называемый как-нибудь наподобие *russian*).

Каждый читатель этой книги сможет получить комплект e^mT_EX'a вместе с русификацией I^AT_EX'a 2_E и I^AT_EX'a 2.09, если вступит в ассоциацию *CyrTUG* — российскую ассоциацию пользователей кириллического

TeX'a (этот комплект называется *CyrTUG-emTeX*)³⁴. Наряду с распространением русификации TeX'a, ассоциация *CyrTUG* снабжает пользователей литературой по TeX'у, проводит ежегодные конференции пользователей TeX'a, где можно обменяться опытом, получить необходимые консультации, и т. п. Кроме того, в ассоциации на регулярной основе действуют консультационные пункты и электронный архив. По поводу адреса ассоциации см. с. 147. Закончим последней фразой из книги [3]: don't delay, write today!

³⁴Русификацию *МTeX'a*, разработанную в Независимом Московском университете, с помощью которой и была набрана настоящая книга, можно получить у Александра Шеня (shen@ium.ipb.ras.ru). Планируется разместить эту русификацию на ftp-сервере na.econ.msu.su

Литература

- [1] L. Lamport. *Л^AT_EX. A Document Preparation System, User's Guide and Reference Manual.* — Addison-Wesley, 1985.
- [2] L. Lamport. *Л^AT_EX. A Document Preparation System, User's Guide and Reference Manual.* — Addison-Wesley, 1994.
- [3] D. E. Knuth. *The T_EXbook*, часть А серии *Computers and Typesetting*. — Addison-Wesley, 1984. Русский перевод: Дональд Е. Кнут. *Все про T_EX*. — Протвино, РДТ_EX, 1993.
- [4] H. Partl, E. Schlegl, I. Hyna. *Л^AT_EX-Kurzbeschreibung*. — Пособие в электронном виде; входит в состав пакета emT_EX, доступного по адресу [ftp.dante.de](ftp://ftp.dante.de), directory `tex-archive/systems/msdos/emtex`
- [5] Х. Партель, Э. Шлегль, И. Хина. *Л^AT_EX: краткое описание*. — Пересказ с немецкого предыдущей версии пособия [4] с дополнениями А. Шеня (shen@ium.ips.ras.ru). Рабочие материалы Независимого Московского университета, 1993. В электронном виде распространяется свободно.
- [6] M. Spivak. *The Joy of T_EX. A gourmet guide to typesetting with the A_MS-T_EXmacro package*. — American Mathematical Society, Providence, RI, 1990. Русский перевод: М. Спивак. *Восхитительный T_EX: руководство по комфорtnому изгтотвлению научных публикаций в пакете A_MS-T_EX*. — М., Мир, 1993.
- [7] M. Goossens, F. Mittelbach, A. Samarin. *The L^AT_EX Companion*. — Addison-Wesley, 1994.
- [8] G. Grätzer. *Math into T_EX. A simple introduction to A_MS-L^AT_EX*. — Birkhäuser, 1993.
- [9] H. Kopka, P. W. Daly. *A Guide to L^AT_EX 2_E. Document Preparation for Beginners and Advanced Users*. — Addison-Wesley, 1995.
- [10] N. Walsh. *Making T_EX work*. — O'Reilly & Associates, 1994.

Предметный указатель

- \ (backslash с пробелом) 18, 91
\" 92
_ 16
\# 16
\! 70, 76
\\$ 16
\% 16
\' 92
 в окружении tabbing 186
\+ 187
\, 76, 87
\[65
\] 65
\(64
\) 64
\{ 16, 30, 60
\| 50
\} 16, 30, 60
\!* 110
\\" 68, 109, 183, 189
 в абзаце 109
 в окружении array 68
 в окружении tabbing 183
 в окружении tabular 189
\- 106
 в окружении tabbing 187
 для предотвращения
 переносов 109
\. 92
\/ 95
\: 76
\; 76
\< 187
\= 92
 в окружении tabbing 182
\^ 92
\` 92
\~ 92
\> 182
\@ 91
\!` 93
\?` 93
\@addtoreset 269
\@afterindentfalse 279
\@afterindenttrue 279
\@begintheorem 304
\@biblabel 308
\@chapapp 277
\@cite 308
\@dotsep 283
\@dottedtocline 282
\@endtheorem 305
\@eqnnum 305
\@evenfoot 291
\@evenhead 291
\@idxitem 308
\@makechapterhead 277
\@makefnmark 307
\@makefntext 307
\@makeschapterhead 278
\@oddfoot 291
\@oddhead 291
\@opargbegintheorem 304
\@pnumwidth 283
\@startsection 273
\@tocrmarg 283

- \AA 93
 \AE 93
 \Acute 339
 \Alph 221
 \Bar 339
 \Bbb (*AMS-TEX*) 316
 \Bbbk 333
 \Biggl 62
 \Biggr 62
 \Bigl 62
 \Bigr 62
 \Box 50
 \Breve 339
 \Bump eq 331
 \Cap 330
 \Check 339
 \Cup 330
 \Ddot 339
 \DeclareMathOperator 339
 \DeclareMathOperator* 339
 \Delta 44
 \Diamond 50
 \Dot 339
 \Downarrow 46
 \EuScript 58
 \Finv 333
 \Game 333
 \Gamma 44
 \Grave 339
 \H 92
 \Hat 339
 \Huge 97
 \Im 50
 \Join 45
 \L 93
 \LARGE 97
 \LaTeX 18
 \LaTeXe 18
 \Lambda 44
 \Large 97
 \Leftarrow 46
 \Leftrightarrow 46
 \Lsh 333
 \O 93
 \OE 93
 \Omega 44
 \P 50
 \Phi 44
 \Pi 44
 \Pr 48
 \Psi 44
 \Re 50
 \Rrightarrow 45
 \Roman 221
 \Rrightarrow 333
 \Rsh 333
 \S 50, 88
 \Sigma 44
 \Subset 331
 \Supset 331
 \TeX 18
 \Theta 44
 \Tilde 339
 \Updownarrow 46
 \Upsilon 44
 \Vdash 331
 \Vec 339
 \Vert 51
 \Vvdash 331
 \Xi 44
 \`{
 в окружении **tabbing** 187
 \a 184, 186
 \aa 93
 \abstractname 155
 \acute 63
 \addcontentsline 281
 \address 146, 345
 \addtocontents 280

- \addtocounter 220
 \addtolength 235
 \ae 93
 \afterpage 170
 \aleph 49
 \alph 221
 \alpha 43
 \amalg 44
 \and 157
 \angle 49
 \appendix 156
 \appendixname 155
 \approx 45
 \approxeq 332
 \arabic 221
 \arccos 47
 \arcsin 47
 \arctan 47
 \arg 46
 \arraycolsep 199
 \arrayrulewidth 199
 \arraystretch 201
 \ast 51
 \asubjclass 346
 \asymp 45
 \atop 66
 запрещена в *AMS-TEX*'е
 340
 \author 157
 в *AMS-TEX*'е 345
 \b 92
 \backepsilon 332
 \backmatter 156
 \backprime 331
 \backsimeq 331
 \backsimeq 331
 \backslash 50, 60
 \bar 63
 \barwedge 330
 \baselineskip 120, 151
 \baselinestretch 124
 \batchmode 41
 \beta 43
 \beth 333
 \between 331
 \bf 20, 96
 в формулах 54
 \bfseries 99
 \bibitem 159
 с необязательным
 аргументом 160
 \bibname 155
 \bigcap 48
 \bigcirc 44
 \bigcup 48
 \biggl 62
 \biggr 62
 \bigl 62
 \bigodot 48
 \bigoplus 48
 \bigotimes 48
 \bigr 62
 \bigskip 121
 \bigskipamount 110
 \bigsqcup 48
 \bigstar 333
 \bigtriangledown 44
 \bigtriangleup 44
 \biguplus 48
 \bigvee 48
 \bigwedge 48
 \binom 340
 \binoppenalty 53, 54
 \blacklozenge 333
 \blacksquare 333
 \blacktriangle 333
 \blacktriangledown 333
 \blacktriangleleft 331
 \blacktriangleright 331
 \bold (*AMS-TEX*) 316
 \boldmath 55
 \bot 50
 \botfigrule 303
 \bottomfraction 301

- \bowtie** 45
\boxdot 330
\boxed 340
\boxminus 330
\boxplus 330
\boxtimes 330
\breve 63
\bullet 44
\bumpeq 331
\c 92
\cal 55
\cap 44
\caption 166
 с необязательным аргументом 168
\cases (Plain T_EX) 317
\cc 147
\ccname 147
\cdot 44
\cdots 31
\centerdot 330
\cfrac 338
\chapter 153
\chaptername 154
\check 63
\checkmark 334
\chi 43
\choose 66
 запрещена в *AMS-L^AT_EX*'е
 340
\circ 44
\circle 176
\circle* 176
\circlearrowleft 332
\circlearrowright 332
\circledR 334
\circledS 333
\circledast 330
\circledcirc 330
\circledddash 330
\cite 159
 с необязательным аргументом 159
\cleardoublepage 119
\clearpage 119, 169
\cline 192
\closing 146
\clubsuit 50
\colon 81, 82
\columnsep 150
\columnseprule 150
\complement 333
\cong 45
\contentsname 155, 286
\coprod 48
\copy 266
\copyright 50, 88
\cos 47
\cosh 47
\cot 47
\coth 47
\cr 316, 317
\csc 47
\cup 44
\curlyeqprec 331
\curlyeqsucc 331
\curlyvee 330
\curlywedge 330
\curraddr 345
\curvearrowleft 333
\curvearrowright 333
\d 92
\dag 50
\dagger 44
\daleth 333
\dasharrow 333
\dashleftarrow 333
\dashrightarrow 334
\dashv 45
\date 157
 в письме 146
\dbinom 340
\dblfigrule 303

- \dblfloatpagefraction 302
 \dblfloatsep 302
 \dbltextfloatsep 302
 \dbltopfraction 302
 \ddag 50
 \ddagger 44
 \ddot 63
 \ddots 68
 \dedicatory 346
 \deg 47
 \delta 43
 \det 48
 \frac 340
 \diagdown 331
 \diagup 331
 \diamond 44
 \diamondsuit 50
 \digamma 333
 \dim 46
 \displaystyle 78
 \div 44
 \divideontimes 330
 \documentclass 20, 143
 \documentstyle 19, 141
 \dot 63
 \doteq 45
 \doteqdot 334
 \dotfill 255
 \dotplus 330
 \dots 88
 в *AMS-TEX*'е 338
 \doublebarwedge 330
 \doublecap 334
 \doublecup 334
 \downarrow 46
 \downdownarrows 332
 \downharpoonleft 333
 \downharpoonright 333
 \ell 50
 \em 94
 \email 345
 \emergencystretch 108, 113
 \empty 100
 \emptyset 49
 \enclname 147
 \endinput 33
 \enlargethispage 120
 \enskip 91
 \ensuremath 212
 \epsilon 43
 \eqcirc 331
 \eqno 52
 \eqref 338
 \eqsim 332
 \eqslantgtr 331
 \eqslantless 331
 \equiv 45
 \eta 43
 \eth 333
 \evensidemargin 150
 \exhyphenpenalty 115
 \exists 49
 \exp 47
 \extrarowheight 205
 \fallingdotseq 331
 \fbox 89
 \fboxrule 247
 \fboxsep 247
 \figurename 155, 166
 \fill 122, 259
 \firsthline 205
 \flat 50
 \floatpagefraction 301
 \floatsep 302
 \flushbottom 125
 \fnsymbol 222
 \footnote 101
 \footnotemark 102
 \footnoterule 306
 \footnotesep 307
 \footnotesize 97
 \footnotetext 102
 \footskip 292
 \forall 49

- \frac 30
 \frak (A\kappa-T\EX) 316
 \framebox 247
 в L\ATEX'e 251
 \frenchspacing 90
 \frontmatter 156
 \frown 45
 \fussy 107
 \gamma 43
 \gcd 48
 \ge 29, 45
 \genfrac 340, 341
 \geq 51
 \geqq 331
 \geqlant 46, 331
 \gets 46
 \gg 45
 \ggg 331
 \ggtr 334
 \gimel 333
 \gnapprox 331
 \gneq 331
 \gneqq 331
 \gnsim 331
 \grave 63
 \gtrdot 330
 \gtreqless 331
 \gtreqqless 331
 \gtrless 331
 \gvertneqq 331
 \hangafter 135
 \hangindent 135
 \hat 63
 \hbar 49
 \hbox 252
 \hdotsfor 336
 \headheight 151, 292
 \headsep 151, 292, 293
 \heartsuit 50
 \height 251
 \hfil 254
 \hfill 255
 \hfuzz 112
 \hhline 203
 \hline 190
 \hoffset 152
 \hom 47
 \hookleftarrow 46
 \hookrightarrow 45
 \phantom 80
 \hrule 138
 \rulefill 255
 \hslash 333
 \hspace 91, 123
 \hspace* 92
 \hss 259
 \huge 97
 \hyphenation 106
 \hyphenpenalty 115
 \i 93
 \iiint 339
 \iiint 339
 \iint 339
 \imath 50
 \in 45
 \index 161
 \indexentry 162
 \indexname 155
 \indexspace 161
 \inf 48
 \infty 49
 \input 32
 \int 49
 \intercal 330
 \intertext 343
 \intextsep 302
 \iota 43
 \it 96
 \item 128
 в окружении theindex 161
 квадратная скобка после
 команды 130
 необязательный аргумент
 129, 132

\itemsep 288
 \itshape 99
 \j 93
 \jmath 50
 \kappa 43
 \ker 46
 \keywords 346
 \kill 183
 \l 93
 \label 26
 \labelenumi 232
 \labelenumii 232
 \labelenumiii 232
 \labelenumiv 232
 \labelitemi 231
 \labelitemii 231
 \labelitemiii 231
 \labelitemiv 231
 \labelsep 287
 \labelwidth 287
 \lambda 43
 \land 51
 \langle 60
 \large 97
 \lastline 205
 \lbrace 51
 \lbrack 51
 \lceil 60
 \ldotp 81, 82
 \ldots 31, 88
 \le 29, 45
 \leaders 255, 266
 \leadsto 46
 \left 30, 59
 \leftarrow 51
 \leftarrowtail 333
 \lefteqn 72, 80, 261
 \leftharpoondown 46
 \leftharpoonup 46
 \leftleftarrows 332
 \leftmargin 287
 \leftmark 294
 \leftrightarrow 46
 \leftrightarrows 333
 \leftrightharpoons 332
 \leftskip 284
 \leftthreetimes 330
 \leq 51
 \leqno 53
 \leqq 331
 \leqslant 46, 331
 \lessapprox 331
 \lessdot 330
 \lesseqgtr 331
 \lesseqqgtr 331
 \lessgtr 331
 \lessim 331
 \lfloor 60
 \lg 46
 \lhd 45
 \lim 48
 \liminf 48
 \limits 49
 \limsup 48
 \line 175
 \linebreak 109
 с необязательным
 аргументом 110
 \linethickness 181
 \listfigurename 155
 \listoffigures 168
 \listoftables 168
 \listparindent 288
 \listtablename 155
 \ll 45
 \llap 174
 \llcorner 334
 \lll 331
 \llless 334
 \ln 46
 \lnapprox 331
 \lneq 331
 \lneqq 331
 \lnot 51

- \lnsim** 331
\log 46
\longleftarrow 46
\longleftrightarrow 46
\longmapsto 46
\longrightarrow 45
\looseness 114
\lor 51
\lowercase 296
\lozenge 333
\lrcorner 334
\ltimes 330
\lvertneqq 331
\magstep 313
\magstephalf 313
\mainmatter 156
\makeatletter 268
\makeatother 268
\makebox 244
 в L^AT_EX'e 250
\makeindex 161
\makelabels 147
\maketitle 156
\maltese 334
\mapsto 46
\marginpar 170
 с необязательным
 аргументом 170
\marginparpush 171
\marginparsep 171
\marginparwidth 171
\markboth 293
\markright 293, 295
\mathbb 57
\mathbf 56
\mathbin 82
\mathcal 56
\mathfrak 57
\mathit 56
\mathop 82
\mathrel 82
\mathrm 56
\mathsf 56
\mathstrut 79
\mathsurround 84, 232, 307
\mathtt 56
\matrix (Plain T_EX) 316
\max 48
\mbox 109, 243
 в формулах 58
 для предотвращения
 переноса 109
 как «пустой текст» на
 странице 119
\mdseries 99
\measuredangle 333
\medskip 121
\medskipamount 110
\mho 50, 333
\mid 45
\min 48
\mit 55
\mod 339
\models 45
\mp 44
\mu 43
\multicolumn 192
\multiput 178, 265
\multlinegap 342
\nLeftarrow 333
\nLeftrightarrow 333
\nRightarrow 333
\nVDash 332
\nVdash 332
\nabla 49
\natural 50
\ncong 332
\neq 45
\nearrow 46
\neg 49
\neq 51
\newcommand 210, 218
\newcounter 220

- | | |
|--------------------------------|----------|
| с необязательным аргументом | 224 |
| \newenvironment | 237 |
| \newenvironment* | 238 |
| \newfont | 311 |
| \newlength | 233 |
| \newpage | 119 |
| \newtheorem | 239 |
| \newtheorem* | 344 |
| \nexists | 333 |
| \ngeq | 332 |
| \ngeqq | 332 |
| \ngeqslant | 332 |
| \ngtr | 332 |
| \ni | 45 |
| \leftarrow | 333 |
| \rightarrow | 333 |
| \leq | 332 |
| \leqq | 332 |
| \leqslant | 332 |
| \less | 332 |
| \mid | 332 |
| \noindent | 117 |
| \nolimits | 49 |
| \newlinebreak | 110 |
| \nonfrenchspacing | 90 |
| \nonstopmode | 41 |
| \nonumber | 71 |
| \nopagebreak | 118 |
| \normalfont | 101 |
| \normalmarginpar | 171 |
| \normalsize | 97 |
| \not | 62 |
| \notag | 342, 343 |
| \notin | 45 |
| \parallel | 332 |
| \prec | 332 |
| \preceq | 332 |
| \rightarrow | 333 |
| \shortmid | 332 |
| \shortparallel | 332 |
| \sim | 332 |
| \subsetneq | 332 |
| \subsetneqq | 332 |
| \succ | 332 |
| \succeq | 332 |
| \supseteq | 332 |
| \supseteqq | 332 |
| \triangleleft | 332 |
| \trianglelefteq | 332 |
| \triangleright | 332 |
| \trianglerighteq | 332 |
| \nu | 43 |
| \numberwithin | 338 |
| \vDash | 332 |
| \vdash | 332 |
| \nwarrow | 46 |
| \circ | 93 |
| \oddsidemargin | 150 |
| \odot | 44 |
| \oe | 93 |
| \of | 315 |
| \of (Plain T _E X) | 316 |
| \oint | 49 |
| \omega | 43 |
| \ominus | 44 |
| \onecolumn | 125 |
| \opening | 146 |
| \oplus | 44 |
| \oslash | 44 |
| \otimes | 44 |
| \oval | 176 |
| с необязательным аргументом | 176 |
| \over | 315 |
| \over (Plain T _E X) | 315 |
| \overbrace | 67 |
| \overleftarrow | 64 |
| \overline | 63 |
| \overrightarrow | 64 |
| \owns | 51 |
| \pagebreak | 119 |
| \pagename | 145 |
| \pagenumbering | 148 |

- \pageref 27
 \pagestyle 148, 292
 \par 117
 \paragraph 153
 \parallel 45
 \parbox 245
 в L^AT_EX'e 251
 с необязательным
 аргументом 246
 \parindent 22
 \parsep 288
 \parshape 136
 \parskip 124
 \part 153
 \partial 49
 \partname 155
 \partopsep 288
 \perp 45
 \phantom 80
 \phi 43
 \pi 43
 \pitchfork 331
 \pm 44
 \pod 339
 \poptabs 186
 \pounds 50, 88
 \prec 45
 \precapprox 332
 \preccurlyeq 331
 \preceq 45
 \precnapprox 331
 \precneqq 331
 \precnsim 331
 \precsim 331
 \prime 49
 \proclaim (Plain T_EX) 318
 \prod 48
 \proofname 344
 \proto 45
 \protect 164, 280
 в аргументе
 \addtocontents 280
 ...
- в аргументе \addcontentsline 282
 в пометке 299
 \ps 147
 \psi 43
 \pushtabs 186
 \put 173
 \qbezier 177
 \qquad 58, 75, 76, 91
 \quad 76, 77, 91
 \r 92
 \raggedbottom 125
 \raggedright 111
 \raisebox 248
 \rangle 60
 \rbrace 51
 \rbrack 51
 \rceil 60
 \ref 27
 в окружении enumerate 131
 ссылка на плавающую
 илюстрацию 166
 ссылка на счетчик,
 определенный
 пользователем 226
 ссылки на раздел
 документа 153
 ссылки на формулы 52
 \refname 75
 \refstepcounter 225
 \relax 139
 \relpenalty 54
 \renewcommand 145, 147, 155, 214
 \renewcommand* 218
 \renewenvironment 238
 \reversemarginpar 171
 \rfloor 60
 \rhd 45
 \rho 43
 \right 30, 59
 \rightarrow 51

- \rightarrowtail** 333
\rightharpoondown 46
\rightharpoonup 46
\righthyphenmin 105
\rightleftarrows 333
\rightleftharpoons 46, 332
\rightmargin 287
\rightmark 294
\rightrightarrows 332
\rightskip 284
\rightsquigarrow 332
\rightthreeetmes 330
\risingdotseq 331
\rlap 260
\rm 21, 96
 в формулах 54
\rmfamily 99
\roman 221
 \roman (*AMS-TEX*) 316
\root 315
\root (Plain T_EX) 316
\rtimes 330
\rule 137
 с необязательным
 аргументом 137
\samepage 119
\savebox 266
\sbox 265
\sc 96
\scriptscriptstyle 78
\scriptsize 97
\scriptstyle 78
\scrollmode 41
\scshape 99
\searrow 46
\sec 47
\section 152, 273
 в *AMS-TEX*'е 346
 с необязательным
 аргументом 152
\section* 153
\sectionmark 295
\setcounter 220
\setlength 234
\setminus 44
\settodepth 235
\settoheight 235
\sf 96
 в формулах 54
\sffamily 99
\sharp 50
\shortmid 332
\shortparallel 332
\shoveleft 342
\shoveright 342
\sigma 43
\signature 146
\sim 45
\simeq 45
\sin 47
\sinh 47
\sl 18, 96
 в формулах 54
\loppy 107, 113
 в L_AT_EX'e 2_E 107, 114
\slshape 99
\small 97
\smallfrown 331
\smallsetminus 330
\smallskip 121
\smallskipamount 110
\smallsmile 331
\smash 80
\smile 45
\spadesuit 50
\special 172
 в пакете emT_EX 319
\specialsection 346
\sphericalangle 333
\sqcap 44
\sqcup 44
\sqrt 31
\sqsubset 45, 331
\sqsubseteq 45

- \sqsupset** 45, 331
\sqsupseteq 45
\square 333
\ss 93
\stackrel 66
\star 44
\stepcounter 225
\strut 139, 140, 201, 293
\subitem 161, 309
\subparagraph 153
\subsection 153
 в *AMS-LATEX*'е 346
\subsectionmark 296
\subset 45
\subseteq 45
\subseteqq 331
\subsetneq 331
\subsetneqq 331
\subsubitem 161, 309
\subsubsection 153
 в *AMS-LATEX*'е 346
\succ 45
\succapprox 332
\succcurlyeq 331
\succeq 45
\succnapprox 331
\succneqq 331
\succnsim 331
\sum 48
\sup 48
\suppressfloats 169
\supset 45
\supseteq 45
\supseteqq 331
\supsetneq 331
\supsetneqq 331
\surd 50
\swarrow 46
\symbol 88
\t 92
\tabcolsep 199
\tablename 155, 167

\tableofcontents 158
 стандартное определение 286
\tan 47
\tanh 47
\tau 43
\tbinom 340
\text 337
\textbf 99
\textfloatsep 302
\textfraction 301
\textheight 151
\textit 99
\textmd 99
\textnormal 101
\textrm 99
\textsc 99
\textsf 99
\textsl 99
\textstyle 78
\texttt 99
\textup 99
\textwidth 149
\tfrac 340
\thanks 157
 в *AMS-LATEX*'е 345
\theindex 309
\theoremstyle 344
\theta 43
\thickapprox 332
\thicklines 181
\thicksim 332
\thinlines 181
\thispagestyle 148
\tilde 63
\times 44
\tiny 97
\title 157
 в *AMS-LATEX*'е 345
\to 45
\tolerance 113
\top 50

\topfigrule	303	\v	92
\topfraction	301	\vDash	331
\topmargin	151	\value	222
\topsep	288	\varDelta	337
\topskip	151	\varGamma	337
\totalheight	251	\varLambda	337
\triangle	49	\varOmega	337
\triangledown	333	\varPhi	337
\triangleleft	44	\varPi	337
\trianglelefteq	331	\varPsi	337
\triangleq	331	\varSigma	337
\triangleright	44	\varTheta	337
\trianglerighteq	331	\varUpsilon	337
\tt	96	\varXi	337
в формулах 54			
\ttfamily	99	\varepsilon	43
\twocolumn	124	\varinjlim	339
\twoheadleftarrow	332	\varkappa	46, 333
\twoheadrightarrow	332	\varliminf	339
\u	92	\varlimsup	339
\uchyph	116	\varnothing	50, 333
\ulcorner	334	\varphi	43
\underbrace	66	\varpi	43
\underline	89	\varprojlim	339
\unitlength	173	\varpropto	331
\unlhd	45	\varrho	43
\unrhd	45	\varsigma	43
\uparrow	46	\varsubsetneq	331
\updownarrow	46	\varsubsetneqq	331
\upharpoonleft	333	\varsupsetneq	331
\upharpoonright	332	\varsupsetneqq	331
\uplus	44	\vartheta	43
\uppercase	296	\vartriangle	331
\upshape	99	\vartriangleleft	331
\upsilon	43	\vartriangleright	331
\upuparrows	332	\vbox	262
\urcorner	334	\vdash	45
\usebox	265	\vdots	68
\usecounter	289	\vec	63
\usepackage	23, 143	\vector	175
с необязательным			
аргументом 144			

- в макроопределении 211
в сносках 134
- \verb*** 134
- \voffset** 152
- \phantom** 80
- \rule** 138, 207
- \vspace** 121
- \vspace*** 122
- \wedge** 44
- \widehat** 63
- \widetilde** 64
- \width** 250
- \wp** 50
- \wr** 44
- \xi** 43
- \leftarrow** 338
- \rightarrow** 338
- \yen** 333
- \zeta** 43
- 11pt** (стилевая опция) 23, 142
- 12pt** (стилевая опция) 23, 142
- a4paper** (стилевая опция) 144
- a5paper** (стилевая опция) 144
- abstract** (окружение)
в *AMS-LATEX*'е 346
- afterpage** (стилевой пакет) 170
- align** (окружение) 342, 343
- align*** (окружение) 343
- aligned** (окружение) 343
- amsart** (класс документов) 144, 335, 345
- amsbook** (класс документов)
144, 335, 345
- amscd** (стилевой пакет) 73, 335
- amsfonts** (стилевой пакет) 23, 57, 330, 335, 336
- AMS-LATEX* 12, 46, 50, 51, 57, 73–75, 318, 330–346
- amsmath** (стилевой пакет) 144, 335
- amsproc** (класс документов)
144, 335, 345
- amssymb** (стилевой пакет) 46, 50, 51, 330, 335
- AMS-LATEX* 11, 73, 315–318
- amsthm** (стилевой пакет) 335
- array** (окружение) 67
- array** (стилевой пакет) 204
- article** (стиль в *LATEX*'е 2.09,
класс документов в
LATEX'е 2 ϵ) 141
- at** (ключевое слово) 313
- at-выражение** 198
- aux-файл** 27
- b5paper** (стилевая опция) 144
- badness** 112
- bmatrix** (окружение) 336
- book** (стиль в *LATEX*'е 2.09,
класс документов в
LATEX'е 2 ϵ) 141
- bottomnumber** (счетчик) 300
- CD** (окружение) 73
- center** (окружение) 24, 126
- СТАН 348
- CyrTUG* 351
- dbltopnumber** (счетчик) 301
- definition** (стиль оформления
теорем в *AMS-LATEX*'е)
344
- depth** (ключевое слово) 139
- depth** (ключевое слово) 261
- description** (окружение) 128
- displaymath** (окружение) 65
- draft** (стилевая опция) 143
- dvi**-драйвер 15
- dvi**-файл 15
- empty** (стиль оформления
страниц) 148

- emTeX** 319, 349
enumerate (окружение) 128, 130
eqnarray (окружение) 70
eqnarray* (окружение) 71
equation (окружение) 52
eufrak (стилевой пакет) 58
euscript (стилевой пакет) 58
executivepaper (стилевая опция) 144
- figure** (окружение) 165
 с необязательным аргументом 166
- figure*** (окружение) 167
- fleqn** (стилевая опция) 143
- flushleft** (окружение) 126
- flushright** (окружение) 126
- gather** (окружение) 342
- gather*** (окружение) 342
- Goossens, M. 12
- headings** (стиль оформления страниц) 148
- height** (ключевое слово) 139
- hhline** (стилевой пакет) 203
- idx**-файл 162
- itemize** (окружение) 128, 129
- lof**-команда 281
- landscape** (стилевая опция) 145
- latexsym** (стилевой пакет) 45, 46, 50
- legalpaper** (стилевая опция) 144
- leqno** (стилевая опция) 143
- letter** (окружение) 146
- letter** (стиль в L^AT_EX'e 2.09, класс документов в L^AT_EX'e 2_ε) 141, 146
- list** (окружение) 288–290
- lof**-файл 168, 280
log-файл 34, 103
longtable (стилевой пакет) 23
lot-файл 168, 280
- makeindex** (программа) 323–329
 стилевой файл 328
- math** (окружение) 64
- matrix** (окружение) 336
- MaxMatrixCols** (счетчик) 337
- METAFONT** 311
- minus** (ключевое слово) 257
- Mittelbach, F 12
- multline** (окружение) 341
- multline*** (окружение) 341
- myheadings** (стиль оформления страниц) 148, 298
- notitlepage** (стилевая опция) 145
- oneside** (стилевая опция) 145
- openany** (стилевая опция) 145
- openright** (стилевая опция) 145
- Overfull 103, 258
 в колонтитуле 153
- paragraph** (счетчик) 225
- picture** (окружение) 172
 вложенные окружения 179
- pk**-файл 311
- Plain TeX 11, 315–318
- plain** (стиль оформления страниц) 148
- plain** (стиль оформления теорем в *AMS-LATEX*'е) 344
- plus** (ключевое слово) 257
- pmatrix** (окружение) 336
- proc** (класс документов) 144, 145
- proof** (окружение) 344

- quotation** (окружение) 126
quote (окружение) 126
- remark** (стиль оформления теорем в *AMS-TEX*'е) 344
- report** (стиль в *LATEX*'е 2.09, класс документов в *LATEX*'е 2ε) 141
- scaled** (ключевое слово) 312
- Schöpf, R. 12
- secnumdepth** (счетчик) 272
- subparagraph** (счетчик) 225
- subsubsection** (счетчик) 225
- tabbing** (окружение) 182
- table** (окружение) 167
- table*** (окружение) 167
- tabular** (окружение) 188–208 с необязательным аргументом 190
- TeXовское приглашение 39
- tfm**-файл 311
- the**-команда 227
- thebibliography** (окружение) 159
- theindex** (окружение) 160
- titlepage** (стилевая опция) 143
- to** (ключевое слово) 253
- toc**-файл 158, 279
- topnumber** (счетчик) 300
- totalnumber** (счетчик) 300
- trivlist** (окружение) 290
- twocolumn** (стилевая опция) 142
- twoside** (стилевая опция) 142
- Underfull** 104, 258 при нехватке клея 254 при печати страницы 125
- verbatim** (окружение) 133
- в макроопределении 211 в сносках 134
- verbatim** (стилевой пакет) 134
- verbatim*** (окружение) 134
- verse** (окружение) 128
- Vmatrix** (окружение) 336
- vmatrix** (окружение) 336
- width** (ключевое слово) 139
- Абзацы** 16, 102
- абзацный отступ 22
- верстка без выравнивания 111
- дополнительный интервал между абзацами 124
- изменение количества строк 114
- неправильной формы 136
- нестандартной формы 134
- неточное выравнивание по правому краю 112
- подавление отступа 117
- сообщения о трудностях при верстке 103, 104
- Автор** документа 157
- Амперсенд** 189
- Аргумент** 23, 57, 93 необязательный 23 перемещаемый 165
- Базисная линия** (*baseline*) 242
- Базовые файлы** 350
- Бинарные операции** 44, 81, 82
- Бинарные отношения** 45, 81, 82
- Биномиальные коэффициенты** 66
- Блок** (*box*) 242
- Блоковые переменные** 264
- Буквы греческие** 43–44 наклонные 55, 337
- Виноградов, Михаил 351

- Вулис, Дмитрий 351
- Глояти, Нана 351
- Группы 20
- Дефис 86
- Диаграммы 71
- Диакритические знаки 92
в окружении `tapping` 184
- Длина
единицы измерения 25
em 26
ex 26
дюйм 25
пика 25
пункт 25
пункт Дидо 25
цицеро 25
параметры 22, 233
присваивание значений 234
с коэффициентом 235
сложение 235
- Доказательство (*АЛС-ЛТЕХ*) 344
- Дроби 30
- Дроби, цепные 78, 338
- Заглавие документа 157
- Заметки на полях 170
- Иллюстрации 165
при наборе в две колонки 167
стандартное название 166
- Индексы 28, 29
- Интеграл 49
двойной, тройной, и т. д. 77, 339
контурный 49
- Интерлиньяж 123
- Кавычки 87
- елочки 87
лапки 87
- Кернинг 87
- Клей 122, 257
бесконечно сжимаемый 259
- Кнут, Дональд 11
- Колонтитулы 291
высота 292
интервал между колонтитулом и текстом 292
передача информации из текста 293
- Команды 17
примитивные 215
пробел после имени 18
со звездочкой 25
хрупкие (fragile) 165
- Комментарии 17
- Коммутативные диаграммы 73
- Корень 31
- Коррекция наклона 95
в L^AT_EX'е 2100
- Лапко, Ольга 351
- Лигатуры 87
- Лидеры 255
в оглавлении 283
использование блоковых переменных 266
- Линейки 137
невидимые 139, 197, 201
в формулах 79
- Лэмпорт, Лесли 11
- Макросы 209
новые окружения 237
с аргументами 216
- Маргиналии 170
- Масштабирование 312
- Матрицы 67
преамбула 68

- Маттес, Эберхард 319
Маховая, Ирина 10
Многоточие 88
 в тексте 88
 в формулах 31
 в *ЛМС-ЛТЭХ*'е 338
- Надстрочные знаки
 в тексте 92
 в формулах 63
 в пакете *amsmath* 339
- Неразрывный пробел ~ 89
- Оглавление 158
 запись текста без номера
 страницы 280
 запись текста с номером
 страницы 281
 модификация оформления
 279
 стандартный заголовок 158
 степень детализации 273
- Ограничители 60
- Окружения 24
 типа «теорема» 239
 в *ЛМС-ЛТЭХ*'е 344
- Операторы типа сумма 48, 81, 82
- Опции стилевые 23, 142
 у пакета 144
- Пакет, стилевой 23
- Переносы
 в словах с дефисом 105
 в словах, начинающихся с
 прописной буквы 116
 двух последних букв 105
 затруднение и запрет 115
 предотвращение в данном
 слово
 с помощью `\mbox` 109
 с помощью `\-` 109
- указание разрешенных мест
 «глобальное» 106
 «локальное» 106
- Перечеркнутые символы 62
- Перечни
 модификация
 заголовков `enumerate` 232
 заголовков `itemize` 231
 нумерованные (`enumerate`)
 130
 общего вида (`list`) 288–290
 примитивные (`trivlist`)
 290
 простейшие (`itemize`) 129
 с заголовками
 (`description`) 133
- Пика 25
- Плавающие иллюстрации
 при наборе в две колонки
 167
 стандартное название 166
- Плавающие таблицы 167
 стандартное название 167
- Подчеркивание 89
- Поля 149
 верхнее и нижнее 151
 левое и правое 150
- Пометки (`marks`) 294
 автоматическое внесение в
 текст 295
- Преамбула документа 19
- Предметный указатель 160
 модификация оформления
 308
 стандартное заглавие 161
- Промежутки
 в формулах 76, 82
 вертикальные 121
 горизонтальные 91, 92
 дополнительные после
 конца предложения 90
- Пункт 25

- Пункт Дидо 25
- Разделы**
- варианты со звездочкой 153
 - модификация оформления 273, 277, 279
 - подавление отступа в первом абзаце 154, 274
 - подавление отступа в первом абзаце главы 279
 - стиль оформления заголовка 275
 - уровень вложенности 272
- Рамки 89, 247, 248, 340
- Режимы *TeX*'а 116
- Самарин, Александр 12, 351
- Скобки 30
 - горизонтальные 66, 67
 - переменного размера 30, 59, 60, 62
 - в *AMSLATEX*'е 337
- Сноски 101
 - к тексту внутри блока 102
 - к титльному листу 157
 - линейка, отделяющая сноски от текста 306
 - оформление номеров 307
 - оформление текста сноски 307
 - пробел между страницей и сносками 306
- Спивак, Майкл 14
- Список иллюстраций 168
- Список литературы 159
 - скобки вокруг номера
 - ссылки 308
 - стандартное заглавие 160
- Список таблиц 168
 - стандартное заглавие 155
- Сравнения по модулю 47
- Степени 28, 29
- Стихи 128
- Страницы**
- запрет разрыва 118
 - глобальный 119
 - изменение размера
 - отдельной полосы 120
 - принудительный разрыв 119
 - при двустороннем наборе 119
 - с выдачей плавающих иллюстраций 119
 - сдвиг как целого 152
 - стиль нумерации 148
 - стиль оформления 148
 - модификация 291
- Стрелки 45, 46, 332–333
 - надпись над стрелкой 66, 74
 - надпись под стрелкой 74
 - надпись сбоку от стрелки 72, 74
- Строки
- жидкие 103
 - равномерное увеличение разреженности 108
 - снятие запретов 107
 - запрет разрыва 110
 - неразрывный пробел 89
 - насильственный разрыв 109
 - с выравниванием 109
- Счетчики 219
 - the*-команда 227
 - выдача на печать 221, 222
 - определенные при начале трансляции 231
 - переподчинение
 - существующего счетчика 269, 338
 - подчинение 224
 - присваивание значений 220
 - сложение 220

- создание 220
 ссылочный префикс 270
 увеличение на шаг 225
- Таблицы**
- абзац в графе 193
 - в пакете *array* 206
 - вертикальные линейки 191
 - горизонтальные линейки 190
 - графы в несколько колонок 192
 - интервал между колонками 199
 - невидимые линейки в строках 201
 - преамбула 188
 - !-выражение (в пакете *array*) 207
 - at-выражение 198
 - символ | 191
 - разбиение преамбулы на колонки 194
 - толщина линеек 199
 - частичные горизонтальные линейки 192
- Табуляция**
- выравнивание по правому краю 186
 - запоминание и вспоминание позиций 186
 - использование позиций 182
 - предварительная установка позиций 183
 - сдвиг первой позиции 187
 - установка позиций 182
- Тангенс 47
- Тире 86
 - длинное (em-dash) 86
 - короткое (en-dash) 86
- Титульный лист 156
- дополнительная информация 157
 оформление вручную 158
 сноски 157
- Точка отсчета 174, 242
- Ударение** 93
- Форматный файл** 349
- Формулы**
- альтернативные обозначения 64
 - включение текста 58
 - внутритечевые 28, 64
 - выключные 28, 65, 143
 - знак препинания после формулы 31
 - надстрочные знаки 63
 - нумерация 52, 53, 143, 269, 305
 - переносы 53, 75
- Цепные дроби 78, 338
- Цитаты 126
- Цицеро 25
- Шень, Александр 14, 351
- Шрифты**
- typewriter* 96, 314
 - в формулах 54
 - капитель 96
 - курсивный 96
 - математический курсив 28
 - наклонный 18, 96
 - полужирный 20, 96
 - прямой светлый (роман) 96, 313
 - рубленый 96
 - рукописный 55
- Штрихи (в формулах) 31, 50, 83, 84

Оглавление

Предисловие

9

I Элементарное введение

11

1.	Общие замечания	11
1.1.	Что такое ТЕХ и ЛАТЕХ	11
1.2.	Достоинства и недостатки	12
1.3.	Литература по ТЕХ'у	14
1.4.	Как проходит работа с системой ЛАТЕХ	14
2.	Основные понятия	15
2.1.	Исходный файл	15
2.2.	Спецсимволы	16
2.3.	Команды и их задание в тексте	17
2.4.	Структура исходного текста	19
2.5.	Группы	20
2.6.	Параметры	22
2.7.	Команды с аргументами	22
2.8.	Окружения	24
2.9.	Звездочка после имени команды	25
2.10.	Единицы длины	25
2.11.	Автоматическая генерация ссылок	26
3.	Набор формул в простейших случаях	28
3.1.	Основные принципы	28
3.2.	Степени и индексы	28
3.3.	Дроби	29
3.4.	Скобки	30
3.5.	Корни	31
3.6.	Штрихи и многоточия	31
3.7.	Функции типа синус	32
4.	Разбиение исходного файла на части	32
5.	Обработка ошибок	34
6.	Как читать книгу дальше?	41

II Как набирать формулы	43
1. Таблицы спецзнаков с комментариями	43
1.1. Операции, отношения и просто значки	43
1.2. Операции с пределами и без	46
1.3. Разное	49
1.4. Какие еще есть символы	51
2. Важные мелочи	51
2.1. Нумерация формул	51
2.2. Переносы в формулах	53
2.3. Смена шрифтов в формуле: $\text{\LaTeX} 2.09$	54
2.4. Смена шрифтов в формуле: $\text{\LaTeX} 2\epsilon$	56
2.5. Включение текста в формулы	58
2.6. Скобки переменного размера	59
2.7. Перечеркнутые символы	62
2.8. Надстрочные знаки	63
2.9. Альтернативные обозначения для математических формул	64
3. Одно над другим	65
3.1. Простейшие случаи	65
3.2. Набор матриц	67
3.3. Набор коммутативных диаграмм в $\text{\LaTeX}'е 2\epsilon$	73
3.4. Переносы в выключных формулах: $\text{\LaTeX} 2.09$	75
4. Тонкая настройка	76
4.1. Пробелы вручную	76
4.2. Размер символов в формулах	77
4.3. Фантомы и прочее	79
4.4. Снова об интервалах в формулах	81
4.5. Дополнительные пробелы вокруг формул	84
III Набор текста	86
1. Специальные знаки	86
1.1. Дефисы, минусы и тире	86
1.2. Кавычки	87
1.3. Многоточие	88
1.4. Прочие знаки	88
1.5. Подчеркивания, рамки	89
2. Промежутки между словами	89
2.1. Неразрывный пробел	89
2.2. Промежутки между предложениями	90
2.3. Установка промежутков вручную	91
3. Диакритические знаки и прочее	92

4.	Смена шрифтов в тексте: \LaTeX 2.09	94
5.	Смена шрифтов в тексте: $\text{\LaTeX}_2\epsilon$	98
6.	Сноски	101
7.	Абзацы	102
7.1.	Переносы	105
7.2.	Команда $\backslash sloppypar$ и параметр $\backslash emergencystretch$	107
7.3.	Ручное управление разрывами строк	108
7.4.	Верстка абзацев без выравнивания и переносов	111
7.5.	Более тонкая настройка	111
8.	Между абзацами	116
8.1.	Понятие о режимах $\text{\TeX}'a$	116
8.2.	Подавление абзацного отступа	117
8.3.	Управление разрывами страниц	118
8.4.	Вертикальные промежутки	121
8.5.	Интерлинияж	123
8.6.	Набор в две колонки	124
8.7.	Заключительные замечания о разрывах страниц и вертикальных интервалах	125
9.	Специальные абзацы	125
9.1.	Цитаты	126
9.2.	Центрирование, ровнение текста по краю	126
9.3.	Стихи	128
9.4.	Перечни	128
9.5.	Буквальное воспроизведение (<i>verbatim</i> , <i>verb</i>)	133
9.6.	Абзацы нестандартной формы	134
10.	Линейки	137
10.1.	Линейки в простейшем виде	137
10.2.	$\text{\TeX}'$ овские команды для генерации линеек	138
10.3.	Невидимые линейки	139
IV	Оформление текста в целом	141
1.	Стили и стилевые опции в $\text{\LaTeX}'e$ 2.09	141
2.	Классы документов и их опции в $\text{\LaTeX}'e$ 2 ϵ	143
3.	Деловые письма на $\text{\LaTeX}'e$	146
4.	Стиль оформления страницы	148
5.	Поля, размер страницы и прочее	149
5.1.	Ширина	149
5.2.	Высота	151
5.3.	Сдвиг страницы как целого	152
6.	Разделы документа	152
6.1.	Команда $\backslash section$	152

6.2.	Какие бывают разделы документа	153
6.3.	Изменение стандартных заголовков	154
6.4.	До и после основного текста	155
7.	Титул, оглавление и пр.	156
7.1.	Титульный лист	156
7.2.	Оглавление	158
7.3.	Список литературы	158
7.4.	Предметный указатель	160
7.5.	Перемещаемые аргументы и хрупкие команды	164
8.	Плавающие иллюстрации и таблицы	165
8.1.	Нештатные ситуации с плавающими объектами в \LaTeX^e 2.09 и в \LaTeX^e 2 ϵ	168
9.	Заметки на полях (маргиналии)	170
V	Псевдорисунки	172
1.	Создание псевдорисунка	172
2.	Отрезки и стрелки	175
3.	Окружности, круги и овалы	176
4.	Кривые (\LaTeX^e 2 ϵ)	177
5.	Дополнительные возможности	178
6.	Параметры оформления псевдорисунка	181
VI	Верстка текста с выравниванием	182
1.	Имитация табулятора	182
1.1.	Элементарные средства	182
1.2.	Более сложные средства	185
2.	Верстка таблиц	188
2.1.	Простейшие случаи	188
2.2.	Более сложные случаи	191
3.	Примеры	195
4.	Дополнительные возможности \LaTeX^e 2 ϵ	203
4.1.	Пересечения линеек	203
4.2.	Расширение возможностей <code>array</code> и <code>tabular</code>	204
VII	Создание новых команд	209
1.	Макроопределения	209
1.1.	Команды без аргументов	209
1.2.	Команды с аргументами	215
1.3.	Новые возможности \LaTeX^e 2 ϵ	218
2.	Счетчики	219

2.1.	Создание счетчиков и простейшие операции с ними	220
2.2.	Отношение подчинения между счетчиками	223
2.3.	Организация автоматических ссылок	226
2.4.	Счетчики, которые определять не надо	231
2.5.	Модификация оформления перечней	231
3.	Параметры со значением длины	233
4.	Создание новых окружений	236
4.1.	Новые окружения: общий случай	236
4.2.	Окружения типа «теорема»	239
VIII Блоки		242
1.	Текст состоит из блоков	242
2.	<i>Л</i> A _T E _X 'овские команды для генерации блоков	243
2.1.	Блоки из строк \acute{y}	243
2.2.	Блоки из абзацев	245
2.3.	Текст в рамке; комбинации блоков	247
2.4.	Сдвиги относительно базисной линии	248
2.5.	Блоки: дополнительные возможности <i>Л</i> A _T E _X 'а 2 _c . .	250
3.	Команда <i>\hbox</i>	252
3.1.	Растяжимые интервалы	253
3.2.	Лидеры	255
3.3.	Клей	257
3.4.	Бесконечно сжимаемые интервалы	259
3.5.	Еще раз о линейках	261
4.	Команда <i>\vbox</i>	262
5.	Блоковые переменные	264
IX Модификация стандартных стилей		267
1.	Общие замечания	267
2.	Снова о счетчиках	268
3.	Разделы документа	272
3.1.	Что нумеровать и что включать в оглавление . .	272
3.2.	Модификация команд, задающих разделы	273
4.	Оглавление, список иллюстраций и прочее	279
5.	Перечни общего вида	286
5.1.	Параметры, влияющие на оформление перечней .	287
5.2.	Окружения <i>list</i> и <i>trivlist</i>	288
6.	Колонтитулы	290
7.	Плавающие объекты	300
8.	Разное	304
8.1.	Теоремы, выключенные формулы	304

8.2. Сноски	306
8.3. Список литературы	308
8.4. Предметный указатель	308
Приложение А. О ТeX'овских шрифтах	311
Приложение Б. ИТeX и другие макропакеты	315
Приложение В. Импорт графики в пакете emT_EX	319
Приложение Г. Программа makeindex	323
1. Простейшие средства	323
2. Тонкости	326
3. Настройка программы makeindex	327
Приложение Д. Шрифты AMS	330
Приложение Е. A_MS-ИT_EX в рамках ИT_EX'a 2_ε	335
1. Матрицы (пакет amsmath)	336
2. Маленькие хитрости (пакет amsmath)	337
3. Дроби и их обобщения (пакет amsmath)	340
4. Выключные формулы (пакет amsthm)	341
5. Работа с теоремами (пакет amsthm)	344
6. A _M S-ИT _E X'овские классы документов	345
Приложение Ж. Откуда взять TeX?	347
1. Электронный дом для TeX'a	348
2. Ассоциации пользователей TeX'a и проблемы русификации	350
Литература	353
Предметный указатель	354